1987

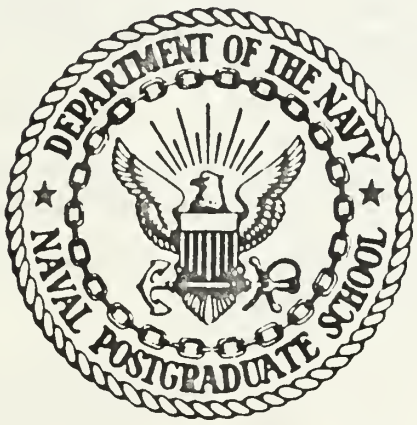
A computer simulation study of an expert system for walking machine motion planning

Goodpasture, Richard Paul

Monterey, California: U.S. Naval Postgraduate School

NPS-52-87-049

NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

A COMPUTER SIMULATION STUDY OF
AN EXPERT SYSTEM FOR
WALKING MACHINE MOTION PLANNING

by

Richard Paul Goodpasture

December 1987

Thesis Advisor: Robert B. McGhee

Prepared for:
Ohio State University
Research Foundation
1314 Kinnear Road
Columbus, OH 43212

DEPARTMENT OF THE NAVY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5100

Rear Admiral R. C. Austin
Superintendent

Kneale T. Marshall
Acting Provost

This thesis prepared in conjunction with research sponsored in part by Ohio State Univ. Research Foundation under RF Project No. 716520.

Released By:

UNCLASSIFIED

CURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION NCLASSIFIED	1b RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release, distribution is unlimited
DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S) NPS-52-87-049	5 MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION val Postgraduate School 6b OFFICE SYMBOL (If applicable) 52	7a NAME OF MONITORING ORGANIZATION Prof. Kenneth Waldron, Dept of Mech Eng'g, Ohio State University
ADDRESS (City, State, and ZIP Code) nterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) 2075 Robinson Laboratory 206 W. 18th Ave Columbus, OH 43210
NAME OF FUNDING SPONSORING ORGANIZATION Ohio State iversity Research Found. 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER RF Project No. 716520 RF Purchase Order No. 496549
ADDRESS (City, State, and ZIP Code) 314 Kinnear Road Columbus, OH 43212	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

TITLE (Include Security Classification) A COMPUTER SIMULATION STUDY OF AN EXPERT SYSTEM FOR WALKING MACHINE MOTION PLANNING

PERSONAL AUTHOR(S)
GOODPASTURE, Richard P.

a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1987 December	103

SUPPLEMENTARY NOTATION

COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Robotics, Walk motions, Adaptive Suspension Vehicle, Robot Motion Planning

ABSTRACT (Continue on reverse if necessary and identify by block number)

A basic algorithm for motion planning of an autonomous robotic vehicle is developed and presented in this study. The algorithm is implemented using simple rules in an expert system shell. This algorithm will allow autonomous vehicles with only rudimentary sensory abilities to navigate from point A to point B with no previous knowledge of the terrain. For this study, the algorithm is implemented with a KEE expert

DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
a NAME OF RESPONSIBLE INDIVIDUAL Robert B. McGhee	22b TELEPHONE (Include Area Code) (408) 646-2095 22c OFFICE SYMBOL 52Mz

D FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office: 1986—606-24.

UNCLASSIFIED

19. Abstract (continued)

system shell on a Texas Instrument Explorer Lisp machine and controls a color graphics computer simulation of the Ohio State University Adaptive Suspension Vehicle.

Approved for public release; distribution is unlimited.

A Computer Simulation Study Of An Expert System For Walking Machine Motion Planning

by

Richard Paul Goodpasture
Lieutenant, United States Navy
B.S., Eastern Kentucky University, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

December 1987

ABSTRACT

A basic algorithm for motion planning of an autonomous robotic vehicle is developed and presented in this study. The algorithm is implemented using simple rules in an expert system shell. This algorithm will allow autonomous vehicles with only rudimentary sensory abilities to navigate from point A to point B with no previous knowledge of the terrain. For this study, the algorithm is implemented with a KEE expert system shell on a Texas Instrument Explorer Lisp machine and controls a color graphics computer simulation of the Ohio State University Adaptive Suspension Vehicle.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. GOALS

The purpose of this study is to investigate a simple motion planning algorithm for navigation of autonomous robotic vehicles. The study will determine whether a motion planning algorithm can be implemented using basic rules within an expert system shell. The expert system shell chosen for this study is the KEE system by Intellicorp [1]. The robotic vehicle that was chosen for the study is the Adaptive Suspension Vehicle (ASV). The ASV is a hexapod walking machine that is currently being evaluated by Ohio State University for natural terrain locomotion [2]. The ASV is a Defense Advanced Research Projects Agency (DARPA) sponsored proof of concept project.

A secondary goal of the study is to develop the communications software necessary for the motion planning algorithm to communicate with the existing ASV computer graphics simulation presented by Lyman [3]. This will allow researchers to test new algorithms using the ASV graphics simulation model rather than the actual machine.

B. THESIS ORGANIZATION

Chapter II provides a brief summary of previous work relating to this study. It includes a discussion on the state of the art of walking machine technology, robotic motion planning, graphical simulations, and expert system shells.

A detailed discussion of the motion planning problem is presented in Chapter III. The discussion includes a description of the ASV, the operator interface with ASV, and the terrain sensing ability of the ASV. Descriptions of the motion planning algorithm

and the hardware and software facilities used in the development of the algorithm are presented in the final sections of this chapter.

A detailed description of the motion planning program's operation is presented in Chapter IV. This includes a complete discussion of the communications software and of the motion planning knowledge base. The final section contains a user's guide, which describes the operation of the program control and display features.

Chapter V is a review of the performance of the motion planning algorithm. It includes examples of typical results, and performance characteristics of the simulation program. The final chapter provides a summary of the research contributions of this study, as well as possible extensions for future research. Portions of the program code are listed in the appendix.

II. SURVEY OF PREVIOUS WORK

A. INTRODUCTION

Research on mobile robots began in the late sixties with a few large projects in the United States and other countries to develop autonomous robots. Research slowed in the seventies, however, as several problems, such as vision, natural language, and motion planning proved harder to solve than expected. As these projects failed to produce all they had promised, interest waned and funding was reduced [4].

Recent years have witnessed important advances in research in image processing, natural language, and motion planning. These advances coupled with major breakthroughs in microprocessor technology, which allow tackling these problems in real time, have sparked renewed interest in all areas of robotics. One specific area of research receiving growing attention involves the problem of robot vehicle locomotion over rough terrain. Much of the work in this area has focused on the design and study of the kinematics and dynamics of multilegged robots [5].

B. WALKING MACHINES

Researchers have been intrigued by legged vehicles or walking machines for the past several decades. This interest grew out observation of the great difference in the mobility of large animals when compared to wheeled or tracked vehicles, and has resulted in extensive research efforts.

Early efforts, such as the General Electric Quadruped [6], represented an important "proof of concept" for walking machines. In the case of the Quadruped, the legs were manually controlled by an operator without the aid of a computer. The many degrees of freedom provided by the legs resulted in such complexity that a highly skilled operator was required, and the vehicle proved difficult to operate for extended periods of time [6]. Advances in microcomputer technology and computer-aided simulation have helped solve this problem and significant progress has been made in recent years. Several promising designs have emerged including: the Perambulating Vehicle (PVII) at Tokyo Institute of Technology, the Carnegie-Mellon University hexapod, the Odetics Inc. ODEX I, and the Adaptive Suspension Vehicle (ASV) developed at the Ohio State University [3].

The PVII was developed in 1980. It is a light weight laboratory model Quadruped, which features one of the first pantograph leg constructions designed specifically to provide leg coordination and energy efficient walking. The PVII is able to probe for footholds and maneuver over obstacles with the aid of tactile foot sensors and a microcomputer mounted near the vehicle [3].

The Carnegie-Mellon hexapod, developed in 1982, is a fully self-contained walking machine that is large enough to carry its operator. Its prime mover is a gasoline engine which provides power to the legs via a set of hydraulic actuators. Motion control is accomplished by an on-board microcomputer which interprets the driver's commands and specifies the correct series of leg movement patterns to be used [3].

Odetics Inc. introduced a commercial design in 1983 called ODEX I. The ODEX I uses a unique circular arrangement of six planar pantograph legs. This arrangement

allows the vehicle to adjust its profile to negotiate narrow passages. The ODEX I is controlled by an operator via a radio or fiber optic link to an on-board supervisory microprocessor. Individual leg control is accomplished by a dedicated low-level microprocessor which receives commands from the supervisory microprocessor [3].

The Adaptive Suspension Vehicle, developed at Ohio State University, is the first computer coordinated legged vehicle designed specifically for operation on natural terrain [2]. The ASV is a hexapod walking machine that is fully self-contained. It is capable of carrying a driver, a 500 pound internal payload, and a complete control and power system in an outdoor environment [3]. The ASV in its current configuration is not an autonomous robot. However, the mechanical and control technologies necessary for unmanned operation are available, and intensive efforts are underway to realize this capability [2]. A simple motion planning algorithm to control the ASV simulation model is presented in this study.

C. ROBOT MOTION PLANNING

Robot motion planning has been one of the central activities in the robotics research community for many years. Two broad categories of motion planning research have emerged. The first is the planning involved to enable a manipulator to avoid collisions in a cluttered workspace [7]. The second concerns planning necessary for mobile robot to move from point A to point B while avoiding obstacles. This study deals entirely with the latter type of motion planning and a brief summary of previous work in this area follows.

Route planning is an important ability of any truly autonomous vehicle. It can be thought of in two basic classes, *long range* and *local* motion planning. The first class,

long range motion planning, generally requires some form of map, since the range of movement is greater than the on-board sensory ability of the vehicle. The second class, local motion planning, is generally restricted to planning movements within an area limited by the scanning range of the vehicle's sensors. [8]

1. Long Range Motion Planning

There are many situations in which long range motion planning requires some degree of optimality. The problem to be solved in this case has been named the *weighted-region* problem [9]. It requires finding a minimum-cost path between a start and a goal which lie in the same cartesian plane. The problem assumes, as a given, that there exists an *area-cost map* that includes the start and goal and the minimum-cost path between them. The area-cost map is comprised of areas of equal *cost-rate* known as *homogeneous-cost* regions. The cost-rate is a measure of cost per unit distance, and is a generic measure in that it could be given in units of time, exposure to danger, fuel expended, or any other appropriate unit of measure. Cost rates are determined only for a specific vehicle and only in terms of location (that is, not in terms of heading or time). Thus, each homogeneous-cost region has a single cost-rate associated with it, and these cost-rates can be used to search for the minimum-cost path. [10]

Various methods have been proposed to find the minimal-cost path. These methods are differentiated by their various control strategies. These strategies are said to be *systematic* if they are *complete* and *non-redundant*. A strategy that is complete is one that will find a solution, if one exists. A strategy that is non-redundant implies that the search will not explore any alternative more than once. *Non-systematic* strategies are those that do not meet the above criteria, and are not suitable for the weighted-region

problem. Two of the simplest systematic search strategies are *depth-first search* and *breadth-first search.* These strategies vary in the manner in which the nodes of the area-cost map are expanded. In breadth-first search, all nodes of the same depth are expanded before the search moves to a greater depth in the map. Depth-first search uses a different approach. In this strategy, a given node will be expanded until the goal is reached or until it can no longer be expanded. When the node can no longer be expanded, the search must *backtrack* to the last node that could be expanded, and continue from there. Either of these strategies can return the first solution they find or heuristics can be added to ensure that an optimal solution is determined. [10]

Richbourg [10] presents an excellent overview of various advanced search strategies to find the minimal-cost path. These are based on the basic systematic strategies discussed above and include *uniform-cost search, best-first search,* and *A* search.*

2. Local Motion Planning

Local motion planning uses the same basic strategies as long range planning. However, the strategies are applied only to an area within scanning range of the vehicle's sensors. In addition, a map of the area may not be available. In this case, one solution is to represent the local environment by a grid. The grid can be filled in as information is provided by on-board sensors and can be used for future planning. This technique is similar to that used by the Stanford Research Institute in their work on the SHAKEY robot [11]. This is the type of elementary planning chosen for this study. A detailed discussion of the algorithm used in the study is presented in Chapter III.

D. GRAPHICAL SIMULATION OF ROBOTIC SYSTEMS

Recent advances in the computer industry have provided a wide variety of options from which to choose graphical displays for robotic systems. Decisions on which option to used must be based on a variety of factors. Questions to be asked include:

- Will the simulation use a monochrome or color monitor?
- Will the simulation use wire-frame or solid figure representation?
- What type of projection and how many dimensions will be used?
- What resolution and what update time is acceptable?
- Is special hardware required?

State of the art graphics systems also offer such options as shading, reflectivity of surfaces, and multiple light sources. Trade-offs must be made, when considering these options, between performance, visual realism, and cost. An effective simulation will involve a compromise between all the options, and will be system dependent [3].

Early simulation models of the ASV by Kwak [12] and Lee [13] concentrated on basic functionality in the display. The vehicles and terrain in these simulations were represented by basic wire-frame figures on monochrome monitors. The first realistic three dimensional simulation of the ASV was created by Lyman [3]. This simulation was written in the C programming language and was implemented on the IRIS-2400 graphics workstation [14]. The IRIS-2400 was chosen a good comprise between state of the art, cost, processing time, and availabilty [3]. The simulation provides a very realistic representation of the ASV and the operator interface. For this reason, this simulation was selected to support the work of this thesis. A more detailed discussion of graphics hardware and software is presented in Chapter III.

E. COMMERCIAL EXPERT SYSTEM SHELLS

Traditional computer technology offers many powerful methods for solving problems that can be clearly and completely defined and have algorithmic solutions. In many cases however, the problem is difficult to define and these methods cannot be applied. In these cases experts are needed to gather and interpret data and select a strategy for solving the problem. *Expert system shells* are a relatively new type of software system designed to augment the decision processes of human experts. This new generation of software has been used in a variety of fields including disease diagnosis, circuit design, planning experiments, and equipment trouble-shooting [1]. A relatively recent development in the robotics research field is the use of expert systems shells to perform the higher level control and planning functions of robots. Although there are many of these expert system shells available, the two most widely used are KEE by Intellicorp [1] and ART by Inference Corporation [15]. Both provide powerful development systems, that are easy to use, provide for rapid prototyping, and have a variety of knowledge representation facilities. They each organize the data into a knowledge base, and have rule systems for rule-based reasoning about the knowledge base. Additionally, each offers a powerful monochrome graphics interface. Due to the availability of KEE on the machines used for this study, and the user friendliness of KEE, it was chosen as the expert system for this study.

F. SUMMARY

This chapter presents background information on previous research relevant to this study. Included are brief surveys of the state of the art of walking machines, and robot motion planning. In addition, there is a section on selection of graphical displays for

robot simulations, and a section discussing the use commercial systems for high level control of robot systems.

The following chapter contains a more detailed discussion of the ASV, and the planning problem to be solved in this thesis.

III. DETAILED PROBLEM STATEMENT

A. INTRODUCTION

The development of an expert system based motion planning algorithm for an autonomous vehicle is the major objective of this study. The second objective is the development of the communication software necessary for the expert system to communicate with the vehicle simulation. For purposes of this study, the vehicle was simulated by an existing graphical simulation of the ASV.

This chapter presents a detailed discussion of the various aspects of the problems presented above. In it is a general discussion of the ASV and of the graphical simulation of the ASV. Also covered are the terrain sensing abilities of the ASV and the motion planning algorithm. The final sections include a description of the simulation facilities used in this study.

B. GENERAL DESCRIPTION OF THE ASV

1. The Vehicle

The Adaptive Suspension Vehicle (ASV) was completed at Ohio State University in 1985. The ASV is the first self-contained, computer coordinated legged vehicle designed to traverse natural terrain. The vehicle is 3.0 meters in height, 5.2 meters in length and weighs approximately 8000 pounds including the driver, the optical scanner mounted on top of the cab, the fuel and hydraulic fluids, and a 500 pound

internal payload. Power for the ASV is provided by a single 900 cc four-cylinder motorcycle engine. This engine drives an energy storage flywheel from which power is distributed to eighteen hydraulic actuator pumps through a series of quill shafts and toothed belts. [2]

On-board processing and control are accomplished by the ASV with an advanced computer system consisting of thirteen Intel single-board computers and two special purpose computers. Three Intel 86/30 computers provide for leg servo control, with each board controlling two legs. One Intel 86/30 and one 80386 board plan trajectories for individual legs. Two 86/30 boards cooperate to realize inertial navigation and safety functions. One 86/30 board monitors operator inputs while two more generate body motion commands and close a body servo loop. All of these processors are linked via a multibus.

A second multibus links the remaining five computers. One of the special purpose computers generates the internal terrain model from optical radar data. Another computer generates an optimal distribution of forces amoung supporting legs and processes force feedback information to assist the body servo computer located in the first multibus cluster. Two Intel 86/30 boards plan leg sequencing and the associated body trajectory. The last computer in this cluster provides a graphics display to the vehicle operator. The two multibus clusters are connected by a parallel data link [2].

Currently, the ASV is controlled by an operator sitting in the cab at the front of the vehicle. The operator can control the vehicle either at a *supervisory* level by selecting body translational and rotational velocities and allowing the vehicle to automatically place the feet, or in a *precision-footing* mode by coordinating the

individual legs. The other various control modes are discussed in detail in [2]. The operator selects the control mode and provides control input to the system via a keypad and joystick located in the cockpit. The joystick is used to provide continuous rate control for longitudinal, lateral, and turning motion. In addition, two thumb operated mini-joysticks provide intermittent rate control for the remaining three degrees of freedom of the body. The operator receives vehicle status information via two CRT displays and a set of LED bar gages mounted in the ceiling of the cab. Changes in the displays and operator interface used for the graphical simulation are discussed in the next section. [2]

2. The Graphical Simulation

The graphical simulation used in this study was developed by Lyman [3]. It incorporates many features that provide a realistic simulation of the ASV including omnidirectional control, automatic body altitude and attitude control, leg motion planning, and body deceleration. Although the simulation models only the kinematics of the ASV, filters were added between operator inputs and vehicle reactions to provide the operator with more realistic vehicle dynamics. [3]

The simulation was developed in a modular form which provides a very flexible environment for studying various control algorithms. While the simulation provides two modes of operation, *forward-wave gait* and *follow-the-leader gait,* only the forward-wave gait was used in this study. In this mode the program's displays and three-axis control inputs are controlled with a mouse and menu system instead of the joystick used in the actual vehicle. [3]

For this study the simulation program's main loop, *walk.c,* and the command subroutine, *driver.c,* were modified to incorporate the necessary communication software needed to receive command inputs from the motion planning algorithm [3]. A more detailed discussion of the hardware and software facilities used in the study is presented in the final sections of this chapter.

C. TERRAIN SENSING

Terrain sensing abilities are a necessity for any truly autonomous vehicle. These abilities can be as elaborate as state of the art radar vision systems, or as simple as basic tactile sensors. These sensors may provide a detailed global view of the terrain, or simply a limited local representation of the terrain. The algorithm presented in this study requires only local sensing of the terrain and a discussion of the ASV's local sensing abilities follows.

The main terrain sensor of the ASV is an optical scanning rangefinder mounted on top of the cab. The scanner uses a GaAlAs laser diode and mechanical mirrors to sweep over the terrain and provide azimuth angle coverage of approximately 40 degrees to either side of the ASV's longitudinal axis and elevation angle coverage of -15 degrees from the horizon to -75 degrees. The range from the scanner to the terrain is determined by transmitting a sinusoidally modulated laser beam and measuring the phase difference between the transmitted and reflected signals. The maximum range of the optical scanner is approximately 10 meters [2].

In addition to the above scanner, the ASV has other sensors to control its legs. These include three tachometers, three potentiometers, and three differential pressure sensors located in the actuators for each leg. These sensors are connected to the leg

computers which use the sensor data to determine velocity, position, and ground reaction force information for each leg. It is the differential pressure detector that can be used as a simple tactile sensor to test the terrain for firm footholds [2]. The sensors discussed above only produce information about the ASV's local environment. However, information from either can be used as an input to the motion planning algorithm presented in the next section, depending on the how the terrain is to be represented internally.

D. THE MOTION PLANNING ALGORITHM

The motion planning algorithm presented in this study provides a simple means by which an autonomous vehicle with only local sensory abilities can navigate from one point to another while avoiding obstacles. The algorithm was developed using simple rules based on observations of how insects and other small animals appear to plan their motions. A discussion of the algorithm's representation of the terrain and a discussion of the rules which form the basis of the algorithm follows.

The internal terrain model for the algorithm is represented by a rectangular grid with nodes which can be marked to indicate the *state* of the node. The nodes can have three possible states: *unexplored, trail,* and *obstacle.* Unexplored implies that the node has not been sensed by any of the vehicle's sensors. A trail node is one which has been used as part of the path to the goal, and is also used for backtracking. A node is represented as an obstacle once one of the vehicle's sensors has determined that the vehicle cannot safely travel to the physical location represented by the node, or once it has been used for backtracking. In addition to the states mentioned above, the terrain model also represents one node as the start node, and can determine if a node is the goal node.

The basic rules for the algorithm were written with several objectives in mind. It was desired that the algorithm find a path to the goal if one existed, and that the algorithm would not search a failed path more than once. The latter goal was met by not allowing the vehicle to cross its own trail, while the first goal was met by using chronological backtracking as used in Prolog. This resulted in a depth-first search algorithm that will for a finite terrain be guaranteed to find a path to the goal if one or more exist. The resulting rules are shown below:

Rule 1:

> If the current node is not the goal node, and if one or more neighboring nodes are unexplored and not an obstacle, move to the unexplored node closest in angle to the goal node and mark the current position as a trail node.

Rule 2:

> If there are no unexplored nodes that are not obstacles, and the current node is not the goal or start node, mark the current node obstacle and backtrack to the previous trail node.

Rule 3:

> If the current node is the goal node, halt and report success.

Rule 4:

> If the current node is the start node, and there are no unexplored nodes that are not obstacle nodes, halt and report failure.

While the above rules are quite simple, the knowledge base to implement them turned out to be suprisingly lengthy. A detailed discussion of the knowledge base used to implement the above algorithm and the graphical representation of the internal terrain model used by the algorithm is presented in Chapter IV.

E. SIMULATION HARDWARE FACILITIES

The computer algorithm presented in this study is designed to operate on any computer system that supports Intellicorp's KEE expert system software. The computer system chosen for this study is the Texas Instrument Explorer Lisp machine [16]. The Explorer system is an advanced, single user workstation. It provides extensive support for development of complex programs, and is a very affordable system for any application requiring symbolic processing, high quality monochrome graphics, and special-purpose processors. The programming environment of the Explorer includes a high-resolution, interactive display, high speed processing using the Lisp language, a ZMACS editor, networking facilities, and a large memory capacity with a sophisticated memory management system. [16]

The simulation that the algorithm controls is designed to run on a Silicon Graphics, Inc. IRIS-2400 workstation. The IRIS systems are state-of-the-art graphics workstations that consist of a Geometry Pipeline, general purpose microprocessor, a raster subsystem, a high-resolution RGB display monitor and a mouse and keyboard. The power of the IRIS comes from the Geometry Pipeline which consists of a series of custom VLSI chip matrix multipliers. It performs matrix transformations, clipping, and scaling of coordinates, and then sends output to the raster subsystem which performs functions such as filling in pixels, shading, depth-cueing, and hidden surface removal. The above functions enable graphics simulations to work in real time which greatly enhance their realism [3].

The above systems are connected by means of an Ethernet network and both use Transmission Control Protocol/Internet Protocol (TCP/IP) as communications software.

A more detailed discussion of the communication facilities of the machines will be presented in Chapter IV. Other software facilities that were used in this study are discussed in the next section.

F. SIMULATION SOFTWARE FACILITIES

The ASV simulation on the IRIS-2400 is written entirely in the C programing language using IRIS graphics library [3]. The required communications software for the IRIS side of the simulation was also written in C, and consists of calls to subroutines which are be linked to the main program. The specific subroutines are discussed in Chapter IV.

The planning algorithm presented in this study was written KEE. KEE uses a frame-based representation of objects and their attributes to form a knowledge base. The attributes can be either descriptive or procedural in form. There are five basic building blocks used to build a frame in KEE. These are *units, slots, slot values, facets,* and *facet values*. The units represent objects in a knowledge base, while slots represent attributes of the units. The values of these attributes are represented by the slot values. Facets are similarly used to describe slots. [1]

The real power of KEE comes from the ability to associate descriptive and procedural attributes directly with the objects in a frame. This is called *object-oriented programming* and provides KEE with modularity and rapid prototyping ability. This object-orientation is a unifying factor in KEE. Rule-based reasoning, access to Lisp, graphics, and deamons are all object-oriented in KEE. KEE provides full access to the power of Lisp through user defined functions called *methods*. A method is a Lisp function which is situated in the knowledge base as the value of a slot. Using methods as

slot values implies that the actions defined become part of the structure of the knowledge base, and enables units with actions to perform to communicate with each other and to execute actions in response to one another. [1]

The object-oriented programming facilities of KEE provided for rapid development of the knowledge base for the planning algorithm and allowed the use of methods to access Lisp functions written to communicate with the IRIS graphics simulation.

G. SUMMARY

This chapter provides a detailed discussion of the problems considered for this study. It includes a general discussion of the ASV and its terrain sensing abilities. Additionally, the motion planning algorithm used in this study is presented, along with the hardware and software facilities used.

IV. DESCRIPTION OF THE SIMULATION

A. INTRODUCTION

This chapter presents a description of the simulation program used in this study. The first section discusses the communications software required for the different parts of the simulation to communicate with each other. The second section provides a detailed look at the knowledge base used for motion planning and introduces the KEE working environment. The final section of the chapter presents the user's guide for the simulation. It contains complete instructions on how to use each of the different program modules for the simulation.

B. COMMUNICATIONS SOFTWARE

The communications software developed in this study consists of two very distinct software packages. One is necessary to run the *server* side of the simulation on the IRIS workstation and is written in C, while the other is required to run the *client* portion of the simulation on the TI Explorer and is written in Lisp.

The software package on the IRIS side, consists of modifications to the main loop, *walk.c,* and the control subroutine, *driver.c [3].* These modifications include the addition of a queuing subroutine which checks for commands from the motion planner and puts any instructions received on a queue. The queue is polled each time through the main loop and if the queue has data, the commands will be executed. The remainder of the communications software on the IRIS side consists of routines developed by Major Ted Barrow at the Naval Postgraduate School.

The communications package on the TI Explorer side of the simulation is built using an instantiation of the *IP::TCP-HANDLER* flavor available in the IP/TCP software package [16]. Additionally, functions and methods are used to provide the overhead required to interface with the IRIS communications software and to simplify the interface with the KEE software. The software package is written to run in the normal Lisp Listener environment of the TI Explorer or in the KEE environment. However, it is necessary to load the software in the environment it is to be used in. Provisions are included in the motion planning knowledge base to load the communications package into the KEE environment. The Lisp code for the communications package is provided in Appendix A.

C. MOTION PLANNING KNOWLEDGE BASE

The motion planning knowledge base presented consists of the units and slots required to implement the basic rules presented in Chapter III, and to provide the graphical display of the terrain model. The knowledge base should not be viewed as a static system, but as an ever-changing environment. Units have been added to the knowledge base or modified as necessary to expand the terrain model or implement new rules. A graph of the knowledge base showing the relationship of the units in the knowledge is shown in Figure 4.1. It is important to note that units in KEE are of two distinct types, *parent* and *child,* and that these two types of units are represented differently on the graph of the knowledge base. Parent links are shown with solid lines and child links are shown with dashed lines. This is important because only the units that are children can receive messages to activate slot values or methods [1]. A discussion of the main units and their slots follows.

Figure 4.1 Graph Of The ASV Knowledge Base

The *COMMANDERS* unit represents a class of units which form the highest level of control in the knowledge base. At present, the *COMMANDERS* unit has only one child, the *ASV-COMMANDER* unit. This unit contains slots which initialize the terrain model and which send messages to other units to perform the motion planning. The knowledge base is written so that other commander units could be added at a latter time if different types of control schemes are to be considered.

The actual motion planning and execution involves several units. The first of these is the *DRIVERS* unit. This unit simulates the sensing abilities of the ASV. The *DRIVERS* unit also has only one child unit, *ASV-DRIVER*. It contains slots which simulate vision inputs, such as relative position and angle to the goal, to the motion planner. The *ASV-DRIVER* unit also contains slots which send messages to the *DRIVING.RULES* unit, once the vision slot has performed its function. The *DRIVING.RULES* unit forms the *rule class* for the knowledge base and contains all of the actual rules necessary to implement the four basic rules from Chapter III. The rules are written in the *tell and ask* language [1]. Each of these rules is checked against the facts in the knowledge base to determine if the conditions of the rule are satisfied. If they are, then the actions required by the rule are executed by sending messages to the appropriate units.

The units that receive messages from the rules are the *VEHICLES* units. The *ASV* vehicle unit has slots which move the vehicle in the internal terrain model and send the messages necessary to move the graphical simulation on the IRIS-2400.

The *NODES* unit represents a class of units which form the internal terrain model. These units have slots for recording the different states of the node and information about its four nearest neighbor nodes. The child units of the *NODES* unit consist of

KEEPICTURE [1] units required to display the terrain model on the TI Explorer CRT. The resulting display is shown in Figure 4.2.

The remaining units concern the communications with the IRIS simulation of the ASV. The *IRIS.COMMS* unit has two child units, *INITIALIZE.COMMS,* and *TALK*. The first of these has slots which load the Lisp code necessary to communicate with the IRIS into the KEE world. The latter contains slots which send appropriate messages to the IRIS simulation to move the ASV.

The entire code for the knowledge base is too extensive to include in this thesis. Therefore the portion of the knowledge base dealing with the display of the terrain model has been omitted. The remainder of the code is included in Appendix A.

D. USER'S GUIDE

The motion planning knowledge base is relatively easy to use, providing the user has a basic understanding of the TI Explorer and the KEE expert system shell. This following presentation assumes this basic understanding.

The user must first verify that the KEE environment has been loaded into the TI. This is accomplished by noting the KEE trademark symbol at the top of the display in the Lisp Listener environment. The KEE shell is then selected by entering SYSTEM-K on the keyboard. Once in the KEE shell, the user will see the display shown in Figure 4.1. The user then selects the *key* shown in Figure 4.1 with the mouse and picks LOAD KNOWLEDGE BASE. At the prompt, *goodpasturer;asv2.u #2* is entered and the knowledge base is loaded into the system. This requires approximately 17 minutes. When asked if KEE.VIEWPORTS are desired, the user should answer yes by clicking the left mouse button, and the necessary viewports to run the simulation will be

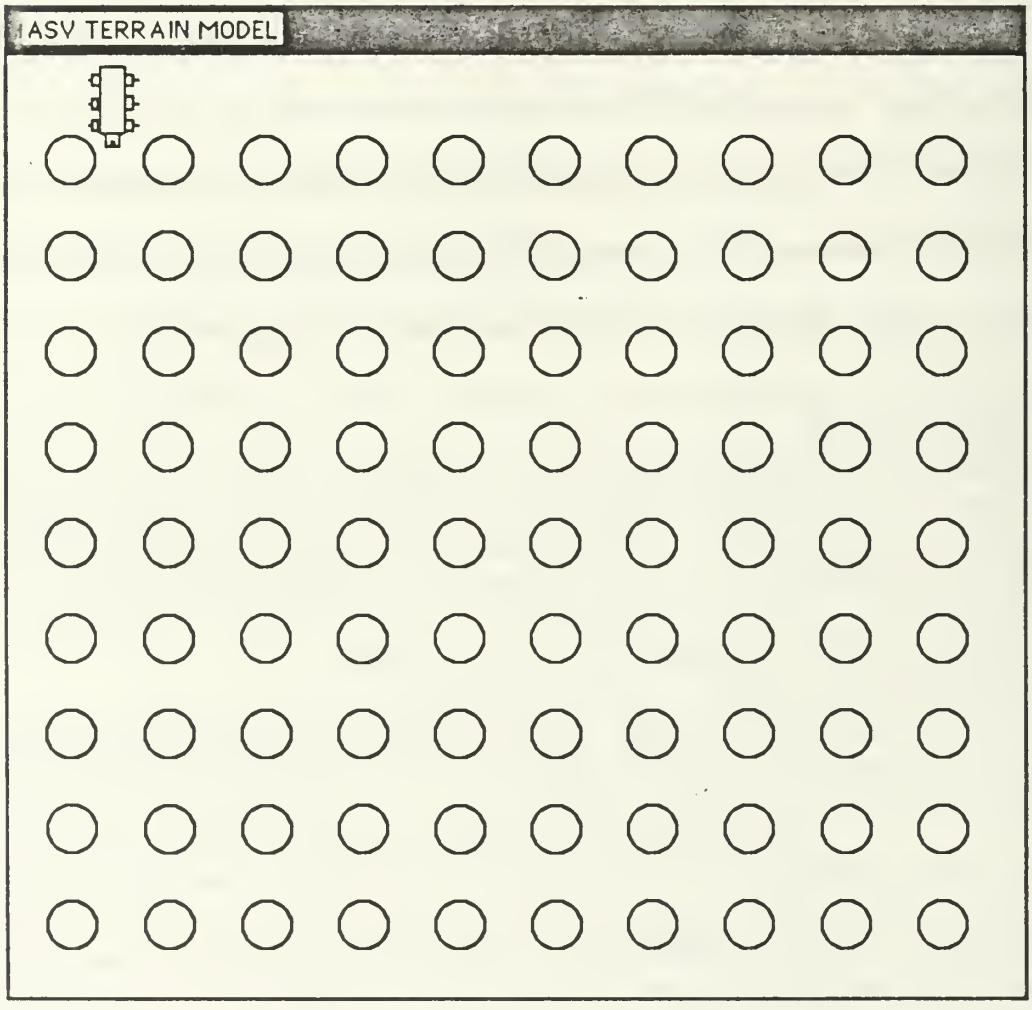


Figure 4.2 Display Of The Internal Terrain Model

displayed. The communication software is loaded by mousing the *LOAD-IRIS.FLAVORS* actuator shown in Figure 4.1. The user must now start the IRIS ASV simulation. This is accomplished by entering the command *"walk"* at the IRIS prompt in the directory containing the simulation [3]. Once the IRIS simulation has started, the user completes the communication link by mousing the *INITIALIZE.COMMS* actuator shown in Figure 4.1. The simulation can now be started by mousing the *START-WALKER* actuator shown in Figure 4.1. Once the ASV has reached the goal, the terrain model can be initialized by mousing the *INITIALIZE-TERRAIN* actuator shown in Figure 4.1 and following prompts. Once the user is finished with the simulation, he must disconnect the communications link by mousing *DISCONNECT.COMMS*. This shuts down the IRIS simulation and the communication software. The user can then continue working in KEE or reboot for the next user by entering META-CONTROL-META-CONTROL-RUBOUT on the keyboard.

E. SUMMARY

This chapter presents a description of the simulation used in this study. The first section describes the communications software used in the study. It includes a discussion of the software used on both the IRIS and TI sides of the simulation. The next section presents a detailed description of the motion planning knowledge base used in the simulation. The final section provides a user's guide to the operation of the simulation. The following chapter presents a description of the simulation performance.

V. SIMULATION PERFORMANCE

A. INTRODUCTION

This chapter presents a performance evaluation of the simulation program developed in this study. It includes a section on the performance of the motion planning program and a section on the performance of the communications software used to link the motion planner to a graphical simulation of the ASV. This review is largely subjective and is based on the author's experience with the operation of the simulation.

B. THE MOTION PLANNING PROGRAM

The motion planning program developed during the study was tested with a variety of different obstacle patterns. The graphical display of the internal terrain model made it very easy to monitor the progress of the program and to determine if the proper decisions were made. In each case, all aspects of the program functioned properly. Several examples of the terrain model displays showing different start patterns and results are included in Appendix B.

The motion planning knowledge base was also run on the SYMBOLICS 3675 Lisp machine. The program also performed properly in each case tested, and ran approximately six times faster. Unfortunately differences in software, and poor documentation, prohibited testing the motion planning program on the SYMBOLICS machine with the IRIS simulation. However, the improved speed on the SYMBOLICS machine indicates that further research into the communications problem would prove fruitful.

C. THE COMMUNICATIONS PROGRAM

The communications software used for the study is still under development, but provides reliable basic communications ability. The package supports sending and receiving of characters, floating point numbers, and integers on both machines. The only limitation noted was the inability to receive negative numbers on the Lisp side. This ability was not used in this study and will have to be added if needed for future research.

The communications package was not implemented on the SYMBOLICS Lisp machine because of basic differences in the IP/TCP software and the lack of documentation. Therefore, no comparison could be made of the total simulation when run on the different machines. However, based on the observations of the motion planning knowledge base when run on the SYMBOLICS, the author believes that the total simulation on the SYMBOLICS and the IRIS would run more efficiently.

D. SUMMARY

This chapter presents an overview of the performance of the motion planning program and the communications package developed during this study. It includes a brief discussion of the abilities and limitations of the simulation. The successful demonstration of these programs indicates that further research in both areas will be profitable.

VI. SUMMARY AND CONCLUSIONS

This thesis demonstrates new techniques for developing robot control and planning programs by means of computer simulations. These techniques involve the use of expert systems to perform the higher level functions of the simulation and the use of specialized computers to perform the more specialized lower level functions of the simulation, such as three dimensional color graphics.

A rule-based motion planning system is developed in this thesis using the KEE expert system shell [1] on a Texas Instrument Explorer Lisp machine [16]. With this system, an autonomous vehicle with no map and only limited sensory abilities can find a path to a known goal while avoiding obstacles.

A communications package is also presented in this study which allows the above expert system to control an existing computer simulation of the Ohio State University ASV robotic vehicle [3]. The techniques developed during this study are intended to be used as generic tools for research in the fields of autonomous vehicles and robotic system simulation.

A. RESEARCH CONTRIBUTIONS

Previous research in the area of robotic system simulation has been concerned with such problems as control, kinematics, and graphical representation of the robot. In some cases, conventional computers were used, resulting in simulations that worked, but with program code that was not understandable at a high level. In other cases, specialized computers such as Lisp machines were used for high level control, but at the expense of

poor graphics displays. There is a need for techniques that will integrate the conventional and specialized computers and allow each to perform the functions of a simulation that they do best.

The simulation presented in this study demonstrates several techniques for maximizing the potential of a robotic simulation. The first part of the simulation involved the development of a rule-based motion planning program which can be used to control an autonomous vehicle with only limited sensory abilities and no map of the environment.

The second part of the simulation involved developing the software package required for the motion planning program to communicate with a graphical simulation on a different computer. The successful demonstration of this ability is an important milestone in robotic research. It will allow future research to use expert systems for high level control of computer simulations on specialized graphics machines or for high level control of conventional computers on actual autonomous vehicles.

Although both of these techniques are demonstrated with a very basic example, the techniques used can be applied to much more complicated examples. These tools provide a notable enhancement to the simulation techniques available for robotics research.

B. RESEARCH EXTENSIONS

The work presented in this study involved several different fields of study, and because of this there are several directions which might be pursued in future research. These fall into three major areas which might be pursued: control of the ASV,

improvement of the motion planning program, and the SYMBOLICS Lisp machine's communication difficulties.

There are several areas in which the control of the ASV simulation can be improved. These include providing the ability to turn the vehicle and the ability to move the vehicle diagonally. Additionally, the ability to start the ASV simulation at a position similar to that of the representation in the internal terrain model could be added. Both of the abilities would add to the realism and practicality of the simulation.

The motion planning program can also be extended in several ways. In particular, eight neighbors could be considered instead of the present four and the vehicle could be allowed to move diagonally. Additionally, some higher level abstractions might be used to reduce the number of actual rules required for the program. Other improvements to be considered include using other rule sets. These rules sets might allow path crossing as an alternative to backtracking if the vehicle could see further ahead and determine that path crossing was more efficient than backtracking. Additionally, rules sets could be implemented that would allow for *trail following* instead of *trail blazing* and allow for shortcuts over known paths. All of these possibilities could be considered along with general streamlining to improve program efficiency.

Another possible extension involves the use of the SYMBOLICS Lisp machine for the motion planning program. Initial estimates indicate that the program runs approximately six times faster than on the TI Explorer. This indicates that efforts in this area would prove fruitful. The additional problem to be solved in this case is the development of a communication link between the SYMBOLICS and the IRIS-2400.

The techniques demonstrated in this thesis open many possibilities for future research. It is hoped that these techniques will provide for more efficient and rapid development of robotic systems.

```
;;;THIS IS PROGRAM IRIS2-FLAVOR.LISP
;;;IT PROVIDES THE NECESSARY OVERHEAD
;;;ON THE TI EXPLORER TO COMMUNICATE
;;;WITH THE IRIS-2400
;;;IT MUST BE LOADED IN THE WORLD IN
;;;WHICH COMMUNICATIONS ARE TO BE USED


(defmacro loopfor (var init test exp1 &optional exp2 exp3 exp4 exp5)
 `(prog ()
       (setq ,var ,init)
       tag
         ,exp1
         ,exp2
         ,exp3
         ,exp4
         ,exp5
         (setq ,var (1+ ,var))
         (if (= ,var ,test) (return t) (go tag))))
;;; handy macro to have in the send message farthur down


(defun convert-number-to-string (n)
 (princ-to-string n))

(defun convert-string-to-integer (str &optional (radix 10))
 (do ((j 0 (+ j 1))
     (n 0 (+ (* n radix) (digit-char-p (char str j) radix))))
    ((= j (length str)) n)))

(defun find-period-index (str)
 (catch 'exit
  (dotimes (x (length str) nil)
    (if (equal (char str x) (char "." 0))
       (throw 'exit x)))))
```

```
(defun get-leftside-of-real (str &optional (radix 10))
 (do  ((j 0 (1+ j))
       (n 0 (+ (* n radix) (digit-char-p (char str j) radix))))
      ((or (null (digit-char-p (char str j) radix)) (= j (length str)))  n)))

(defun get-rightside-of-real (str &optional (radix 10))
 (do ((index (1+ (find-period-index str)) (1+ index))
      (factor 0.10 (* factor 0.10))
      (n 0.0 (+ n (* factor (digit-char-p (char str index) radix)))))
     ((= index (length str)) n )))

(defun convert-string-to-real (str &optional (radix 10))
 (+ (float (get-leftside-of-real str radix)) (get-rightside-of-real str radix)))


(defvar *tcp-handler1* (send ip::*tcp-handler* :get-port))
(defvar *tcp-handler2* (send ip::*tcp-handler* :get-port))

(defvar *iris1-port1* 1027)                ; this is the send port
(defvar *iris1-port2* 1026)                ; this is the receive port
(defvar *iris1-address* 3221866502)            ; tcp-ip or internet address
(defvar *iris2-address* 3221866504)            ; look in network configuration
(defvar *unix1-address* 3221866505)
(defvar *unix2-address* )

(defun choose-iris(x)                      ; iris2 is default
 (cond ((equal x 'iris1) (defvar *dest-address* *iris1-address*))
       ((equal x 'iris3) (defvar *dest-address* *iris3-address*))
       (t          (defvar *dest-address* *iris2-address*))))

(defflavor conversation-with-iris ((talking-port-number    *iris1-port1*)
                                   (listening-port-number  *iris1-port2*)
                                   (talking-port       *tcp-handler1*)
                                   (listening-port     *tcp-handler2*)
                                   (destination        *dest-address*)
                                  )
                                  ()
                                  :gettable-instance-variables
```

```
                    :settable-instance-variables
                    :initable-instance-variables)

(defmethod (conversation-with-iris :initiate-the-conversation) ()
 (progn
  (send talking-port :open
     :active                ; tcp will begin the procedure to establish
                            ; connection  (default  vs :passive)
     talking-port-number    ; port number of destination host
     destination            ; machine name or address if blank and
                            ; in :passive mode local machine waits for
                            ; connection
     30)                    ; set max seconds  before read request times out

  (send listening-port :open
     :active                        ;:passive
     listening-port-number
     destination
     30)
 '"A conversation with the iris machine has been established"))

(defmethod (conversation-with-iris :re-use-iris) ()
  (setq *tcp-handler1* (send ip::*tcp-handler* :get-port)
        *tcp-handler2* (send ip::*tcp-handler* :get-port)
        talking-port *tcp-handler1*
        listening-port *tcp-handler2*))

(defmethod (conversation-with-iris :get-an-object-from-iris) ()
  (let* ((typebuffer   " ")
         (lengthbuffer "    ")
         (buffer       " ")
         (buffer-length 1))
    (progn
      (send listening-port :receive
          typebuffer
```

```
            buffer-length
            30
            :wait)

        (send listening-port :receive
            lengthbuffer
            4
            30
            :wait)

        (setq buffer-length (convert-string-to-integer lengthbuffer))

;       (terpri)
;       (princ "type to receive= ") (princ typebuffer)
;       (terpri)

        (setq buffer (make-string buffer-length :initial-element (character 32)))


        (send listening-port :receive
            buffer
            buffer-length
            30
            :wait)

;       (terpri)
;       (princ "data  received= ") (princ buffer)
;       (terpri)

        (cond ((equal typebuffer "I") (convert-string-to-integer buffer))
            ((equal typebuffer "R") (convert-string-to-real    buffer))
            ((equal typebuffer "C") buffer)
            (t nil)))))

(defvar *step-var* 0)

(defmethod (conversation-with-iris :send-iris-an-object) (object)

  (let* ((buffer (cond
```

```
            ((equal (type-of object) 'bignum) (convert-number-to-string object))
            ((equal (type-of object) 'fixnum) (convert-number-to-string object))
            ((equal (type-of object) 'float) (convert-number-to-string object))
            ((equal (type-of object) 'string) object)
            (t "error")))

        (buffer-length  (length buffer))

        (typebuffer     (cond ((equal (type-of object) 'bignum) "I")
                              ((equal (type-of object) 'fixnum) "I")
                              ((equal (type-of object) 'float) "R")
                              ((equal (type-of object) 'string) "C")
                              (t "C")))

        (lengthbuffer   (convert-number-to-string buffer-length)))

    (progn

;       (terpri)
;       (princ "type sent = ") (princ typebuffer)
;       (terpri)

       (send talking-port :send
            typebuffer
            1
            nil
            nil)


       (if (= (length lengthbuffer) 4)
            (send talking-port :send
                 lengthbuffer
                 4
                 nil
                 nil)
```

```
        (progn
          (loopfor *step-var* (length lengthbuffer) 4
            (send talking-port :send "0" 1 nil nil))

;           (terpri)
;           (princ "length sent ") (princ lengthbuffer)
;           (terpri)

            (send talking-port :send lengthbuffer (length lengthbuffer) nil nil)
          ))

;     (terpri)
;     (princ "data sent= ") (princ buffer)
;     (terpri)
;     (princ "data length=  ") (princ buffer-length)
;     (terpri)

      (send talking-port :send
          buffer
          buffer-length
          t
          nil))))

(choose-iris 2)

(defmethod (conversation-with-iris :stop-talking-to-iris) ()
  (progn (send talking-port :close) (send listening-port :close)))

(setq talk (make-instance 'conversation-with-iris))

(defun start-con () (send talk :initiate-the-conversation))

(defun end-con () (send talk :stop-talking-to-iris))

(defun restart () (send talk :re-use-iris))

(defun get_data ()
   (send talk :get-an-object-from-iris))

(defun send_float (float)
```

```
  (send talk :send-iris-an-object float))

(defun send_string (string)
  (send talk :send-iris-an-object string))
(defun ax () (send talk :send-iris-an-object "e"))

(defun ay () (send talk :send-iris-an-object "n"))

(defun f () (send talk :send-iris-an-object "f"))

(defun r () (send talk :send-iris-an-object "r"))

(defun h () (send talk :send-iris-an-object "h"))

(defun s () (send talk :send-iris-an-object "z"))
(defun dx () (send talk :send-iris-an-object "w"))
(defun dy () (send talk :send-iris-an-object "s"))
(defun rl() (send talk :send-iris-an-object "rl"))

(defun rr () (send talk :send-iris-an-object "rr"))
(defun start-con () (send talk :initiate-the-conversation))
(defun end-con () (send talk :stop-talking-to-iris))
```

```
;;; -*- Mode:Common-Lisp; Syntax:Common-lisp; Package:KEE; Base:10. -*-

(ASV

("goodpasturer" "9-10-87 13:49:01" "goodpasturer" "12-3-87 10:52:05")
 NIL

(KNOWLEDGEBASES)
 NIL
 ()
 (
(DELETE.KB ((BEFORE (BEFORE-AI3-KB-DELETE SELF)) (BEFORE (BEFORE-KP-KB-DELETE SELF))))

(KBMETHODFILE ("ASV"))

(KBSIZE (61))

(KEE.DEVELOPMENT.VERSION.NUMBER (0))

(KEE.MAJOR.VERSION.NUMBER (3))

(KEE.MINOR.VERSION.NUMBER ("05"))

(KEE.PATCH.VERSION.NUMBER (73))

(KEEVERSION (KEE3.0))

(LOAD.KB ((AFTER (AFTER-AI3-KB-LOAD SELF)) (AFTER (AFTER-KP-KB-LOAD SELF))))))

(NODES
 ("goodpasturer" "12-3-87 10:39:07" "goodpasturer" "12-3-87 10:51:36")
 ((ENTITIES GENERICUNITS))
 (TERRAIN2 (CLASSES GENERICUNITS))
 NIL
 ((EAST (NIL))
  (EAST.PATH NIL NIL ((ONE.OF CLEAR OBSTRUCTED EXPLORED)) NIL
```

```
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
 (FIND.EAST
((LAMBDA (THISUNIT)
  (PUT.VALUE THISUNIT
        'EAST
        (PICTURE-AT-POSITION-IN-VIEWPORT 'TANK-WORLD
                        (ADD-POSITION-TO-POSITION (FIND-POSITION-IN-VIEWPORT
                                        THISUNIT
                                        'TANK-WORLD)
                                       (GET.VALUE
                                        'CIRCLE.2
                                        'POSITION))))))
METHOD (#[Unit: METHOD KEEDATATYPES]))
 (FIND.NORTH
((LAMBDA (THISUNIT)
  (PUT.VALUE THISUNIT
        'NORTH
        (PICTURE-AT-POSITION-IN-VIEWPORT 'TANK-WORLD
                        (ADD-POSITION-TO-POSITION (FIND-POSITION-IN-VIEWPORT
                                        THISUNIT
                                        'TANK-WORLD)
                                       (GET.VALUE
                                        'CIRCLE.11
                                        'POSITION))))))
METHOD (#[Unit: METHOD KEEDATATYPES]))
 (FIND.SOUTH
((LAMBDA (THISUNIT)
  (PUT.VALUE THISUNIT
        'SOUTH
        (PICTURE-AT-POSITION-IN-VIEWPORT 'TANK-WORLD
                        (SUBTRACT-POSITION-FROM-POSITION (FIND-POSITION-IN-VIEWPORT
                                        THISUNIT
                                        'TANK-WORLD)
                                       (GET.VALUE
                                        'CIRCLE.11
```

```
                                        'POSITION))))
  ))
METHOD (#[Unit: METHOD KEEDATATYPES]))
  (FIND.WEST
((LAMBDA (THISUNIT)
  (PUT.VALUE THISUNIT
        'WEST
        (PICTURE-AT-POSITION-IN-VIEWPORT 'TANK-WORLD
                        (SUBTRACT-POSITION-FROM-POSITION (FIND-POSITION-IN-VIEWPORT
                                        THISUNIT
                                        'TANK-WORLD)
                                       (GET.VALUE
                                        'CIRCLE.2
                                        'POSITION))))
  ))
METHOD (#[Unit: METHOD KEEDATATYPES]))
  (NORTH (NIL))
  (NORTH.PATH NIL NIL ((ONE.OF CLEAR OBSTRUCTED EXPLORED)) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  (PREVIOUS.DIRECTION)
  (PREVIOUS.POSITION NIL NIL NIL NIL ((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  (SOUTH (NIL))
  (SOUTH.PATH NIL NIL ((ONE.OF CLEAR OBSTRUCTED EXPLORED)) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  (WEST (NIL))
  (WEST.PATH NIL NIL ((ONE.OF CLEAR OBSTRUCTED EXPLORED)) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  )
  ())

(TWO.CHOICE.MOVE.AWAY.SOUTH
  ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:14")
  NIL
  (ASV-DRIVER.RULES)
  NIL
```

```
()
((EXTERNAL.FORM
((IF (?VEHICLE IS IN CLASS VEHICLES)
   (THE DRIVER OF ?VEHICLE IS ?DRIVER)
   (THE RELATIVE.LOCATION.X OF ?DRIVER IS EAST.OF.GOAL)
   (CANT.FIND (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
   (((THE RELATIVE.LOCATION.Y OF ?DRIVER IS SOUTH.OF.GOAL) AND
    (CANT.FIND (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)))
    OR (THE RELATIVE.LOCATION.Y OF ?DRIVER IS AT.GOAL.Y))
   (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
   THEN
   (LISP (UNITMSG ?VEHICLE 'MOVE.SOUTH)))))
 (MAKE.AND.WORLD? (NIL))
 (PARSE.ERRORS)
 (RULE.TYPE (SAME.WORLD.ACTION))
 ))

(TWO.CHOICE.MOVE.AWAY.NORTH
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:13")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 ()
 ((EXTERNAL.FORM
((IF (?VEHICLE IS IN CLASS VEHICLES)
   (THE DRIVER OF ?VEHICLE IS ?DRIVER)
   (THE RELATIVE.LOCATION.X OF ?DRIVER IS WEST.OF.GOAL)
   (CANT.FIND (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
   (((THE RELATIVE.LOCATION.Y OF ?DRIVER IS NORTH.OF.GOAL) AND
    (CANT.FIND (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)))
    OR (THE RELATIVE.LOCATION.Y OF ?DRIVER IS AT.GOAL.Y))
   (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
   THEN
   (LISP (UNITMSG ?VEHICLE 'MOVE.NORTH)))))
 (MAKE.AND.WORLD? (NIL))
```

```
 (PARSE.ERRORS)
 (RULE.TYPE (SAME.WORLD.ACTION))
 ))

(TWO.CHOICE.MOVE.AWAY.WEST
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:12")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 0
 ((EXTERNAL.FORM
((IF (?VEHICLE IS IN CLASS VEHICLES)
   (THE DRIVER OF ?VEHICLE IS ?DRIVER)
   (((THE RELATIVE.LOCATION.X OF ?DRIVER IS WEST.OF.GOAL) AND
    (CANT.FIND (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)))
    OR (THE RELATIVE.LOCATION.X OF ?DRIVER IS AT.GOAL.X))
   (THE RELATIVE.LOCATION.Y OF ?DRIVER IS SOUTH.OF.GOAL)
   (CANT.FIND (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
   (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
   THEN
   (LISP (UNITMSG ?VEHICLE 'MOVE.WEST)))))
 (MAKE.AND.WORLD? (NIL))
 (PARSE.ERRORS)
 (RULE.TYPE (SAME.WORLD.ACTION))
 ))

(TWO.CHOICE.MOVE.AWAY.EAST
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:12")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 0
 ((EXTERNAL.FORM
((IF (?VEHICLE IS IN CLASS VEHICLES)
   (THE DRIVER OF ?VEHICLE IS ?DRIVER)
```

```
    (((THE RELATIVE.LOCATION.X OF ?DRIVER IS EAST.OF.GOAL) AND
     (CANT.FIND (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)))
     OR (THE RELATIVE.LOCATION.X OF ?DRIVER IS AT.GOAL.X))
    (THE RELATIVE.LOCATION.Y OF ?DRIVER IS NORTH.OF.GOAL)
    (CANT.FIND (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
    THEN
    (LISP (UNITMSG ?VEHICLE 'MOVE.EAST)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(BACK.TRACK
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:11")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 ()
 ((EXTERNAL.FORM
 ((IF (?VEHICLE IS IN CLASS VEHICLES)
    (THE DRIVER OF ?VEHICLE IS ?DRIVER)
    (CANT.FIND (THE RELATIVE.LOCATION OF ?DRIVER IS AT.GOAL))
    (CANT.FIND (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (CANT.FIND (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (CANT.FIND (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (CANT.FIND (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    THEN
    (LISP (UNITMSG ?VEHICLE 'BACK.TRACK)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(MOVE.NORTH.AWAY
```

```
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:09")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 0
 ((EXTERNAL.FORM
((IF (?VEHICLE IS IN CLASS VEHICLES)
    (THE DRIVER OF ?VEHICLE IS ?DRIVER)
    (CANT.FIND (THE RELATIVE.LOCATION OF ?DRIVER IS AT.GOAL))
    (CANT.FIND (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (CANT.FIND (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (CANT.FIND (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
    THEN
    (LISP (UNITMSG ?VEHICLE 'MOVE.NORTH)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(MOVE.SOUTH.AWAY
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:09")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 0
 ((EXTERNAL.FORM
((IF (?VEHICLE IS IN CLASS VEHICLES)
    (THE DRIVER OF ?VEHICLE IS ?DRIVER)
    (CANT.FIND (THE RELATIVE.LOCATION OF ?DRIVER IS AT.GOAL))
    (CANT.FIND (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (CANT.FIND (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (CANT.FIND (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
    THEN
```

```
    (LISP (UNITMSG ?VEHICLE 'MOVE.SOUTH)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(MOVE.WEST.AWAY
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:08")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 0
 ((EXTERNAL.FORM
 ((IF (?VEHICLE IS IN CLASS VEHICLES)
    (THE DRIVER OF ?VEHICLE IS ?DRIVER)
    (CANT.FIND (THE RELATIVE.LOCATION OF ?DRIVER IS AT.GOAL))
    (CANT.FIND (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (CANT.FIND (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (CANT.FIND (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
    (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
    THEN
    (LISP (UNITMSG ?VEHICLE 'MOVE.WEST)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(MOVE.EAST.AWAY
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:07")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 0
 ((EXTERNAL.FORM
 ((IF (?VEHICLE IS IN CLASS VEHICLES)
```

```
   (THE DRIVER OF ?VEHICLE IS ?DRIVER)
   (CANT.FIND (THE RELATIVE.LOCATION OF ?DRIVER IS AT.GOAL))
   (CANT.FIND (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
   (CANT.FIND (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
   (CANT.FIND (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))
   (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
   THEN
   (LISP (UNITMSG ?VEHICLE 'MOVE.EAST)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(MOVE.WEST.TO.GOAL
  ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:07")
  NIL
  (ASV-DRIVER.RULES)
  NIL
  ()
  ((EXTERNAL.FORM
 ((IF (?VEHICLE IS IN CLASS VEHICLES)
   (THE DRIVER OF ?VEHICLE IS ?DRIVER)
   (THE RELATIVE.LOCATION.X OF ?DRIVER IS EAST.OF.GOAL)
   ((THE BEST.CHOICE OF DIRECTIONS IS EAST.WEST) OR
    ((THE BEST.CHOICE OF DIRECTIONS IS NORTH.SOUTH) AND
     (((THE RELATIVE.LOCATION.Y OF ?DRIVER IS NORTH.OF.GOAL) AND
       (CANT.FIND (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)))
      OR
      ((THE RELATIVE.LOCATION.Y OF ?DRIVER IS SOUTH.OF.GOAL) AND
       (CANT.FIND (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))))))
   (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
   THEN
   (LISP (UNITMSG ?VEHICLE 'MOVE.WEST)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
```

```
 (RULE.TYPE (SAME.WORLD.ACTION))
 ))

(MOVE.SOUTH.TO.GOAL
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:06")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 ()
 ((EXTERNAL.FORM
((IF (?VEHICLE IS IN CLASS VEHICLES)
   (THE DRIVER OF ?VEHICLE IS ?DRIVER)
   (THE RELATIVE.LOCATION.Y OF ?DRIVER IS NORTH.OF.GOAL)
   ((THE BEST.CHOICE OF DIRECTIONS IS NORTH.SOUTH) OR
    ((THE BEST.CHOICE OF DIRECTIONS IS EAST.WEST) AND
     (((THE RELATIVE.LOCATION.X OF ?DRIVER IS EAST.OF.GOAL) AND
       (CANT.FIND (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)))
      OR
      ((THE RELATIVE.LOCATION.X OF ?DRIVER IS WEST.OF.GOAL) AND
       (CANT.FIND (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))))))
   (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
   THEN
   (LISP (UNITMSG ?VEHICLE 'MOVE.SOUTH)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(MOVE.NORTH.TO.GOAL
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:05")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 ()
 ((EXTERNAL.FORM
```

```
((IF (?VEHICLE IS IN CLASS VEHICLES)
    (THE DRIVER OF ?VEHICLE IS ?DRIVER)
    (THE RELATIVE.LOCATION.Y OF ?DRIVER IS SOUTH.OF.GOAL)
    ((THE BEST.CHOICE OF DIRECTIONS IS NORTH.SOUTH) OR
     ((THE BEST.CHOICE OF DIRECTIONS IS EAST.WEST) AND
      (((THE RELATIVE.LOCATION.X OF ?DRIVER IS EAST.OF.GOAL) AND
        (CANT.FIND (THE WEST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)))
       OR
       ((THE RELATIVE.LOCATION.X OF ?DRIVER IS WEST.OF.GOAL) AND
        (CANT.FIND (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))))))
    (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
    THEN
    (LISP (UNITMSG ?VEHICLE 'MOVE.NORTH)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(MOVE.EAST.TO.GOAL
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "12-3-87 10:34:05")
 NIL
 (ASV-DRIVER.RULES)
 NIL
 0
 ((EXTERNAL.FORM
((IF (?VEHICLE IS IN CLASS VEHICLES)
    (THE DRIVER OF ?VEHICLE IS ?DRIVER)
    (THE RELATIVE.LOCATION.X OF ?DRIVER IS WEST.OF.GOAL)
    ((THE BEST.CHOICE OF DIRECTIONS IS EAST.WEST) OR
     ((THE BEST.CHOICE OF DIRECTIONS IS NORTH.SOUTH) AND
      (((THE RELATIVE.LOCATION.Y OF ?DRIVER IS NORTH.OF.GOAL) AND
        (CANT.FIND (THE SOUTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)))
       OR
       ((THE RELATIVE.LOCATION.Y OF ?DRIVER IS SOUTH.OF.GOAL) AND
        (CANT.FIND (THE NORTH.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR))))))
```

```
   (THE EAST.PATH OF (THE NODE.LOC OF ?VEHICLE) IS CLEAR)
   THEN
   (LISP (UNITMSG ?VEHICLE 'MOVE.EAST)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(DIRECTIONS
 ("goodpasturer" "9-10-87 14:09:42" "goodpasturer" "11-12-87 13:31:48")
 NIL
 ((ENTITIES GENERICUNITS))
 NIL
 ()
 ((BEST.CHOICE (NORTH.SOUTH) NIL ((ONE.OF EAST.WEST NORTH.SOUTH)) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
 ))

(KEEPICTURE.INSTANCES
 ("goodpasturer" "9-10-87 13:52:38" "goodpasturer" "12-2-87 13:17:40")
 NIL
 ((ENTITIES GENERICUNITS))
 "A dummy unit to group all KEEpicture instances together in a KBgraph."
 ()
 ((DELETE-ALL-IMAGES! (AI3-DELETE-ALL-IMAGES) METHOD (#[Unit: METHOD KEEDATATYPES]))
  (GRAPH.ALL.PICTURES! (KP-GRAPH) METHOD (#[Unit: METHOD KEEDATATYPES]))
  (KEEPICTURE.INSTANCES
(#[Unit: BOX-STRING-DISCONNECT.COMMS-OF-TALK.5 ASV (COMPACT.UNIT)]
 #[Unit: INVISIBLE.PICTURE.11 ASV (COMPACT.UNIT)]
 #[Unit: VIEWPORT-DISCONNECT.COMMS-OF-TALK.5 ASV (COMPACT.UNIT)]
 #[Unit: BOX-STRING-INITIALIZE-TERRAIN-OF-ASV-COMMANDER.4 ASV (COMPACT.UNIT)]
 #[Unit: INVISIBLE.PICTURE.9 ASV (COMPACT.UNIT)]
 #[Unit: VIEWPORT-INITIALIZE-TERRAIN-OF-ASV-COMMANDER.4 ASV (COMPACT.UNIT)]
 #[Unit: BOX-STRING-START-WALKER-OF-ASV-COMMANDER.3 ASV (COMPACT.UNIT)]
 #[Unit: INVISIBLE.PICTURE.8 ASV (COMPACT.UNIT)]
```

```
 #[Unit: VIEWPORT-START-WALKER-OF-ASV-COMMANDER.3 ASV (COMPACT.UNIT)]
 #[Unit: BOX-STRING-INITIALIZE.COMMS-OF-INITIALIZE.COMMS.2 ASV (COMPACT.UNIT)]
 #[Unit: INVISIBLE.PICTURE.7 ASV (COMPACT.UNIT)]
 #[Unit: VIEWPORT-INITIALIZE.COMMS-OF-INITIALIZE.COMMS.2 ASV (COMPACT.UNIT)]
 #[Unit: BOX-STRING-LOAD.IRIS-FLAVORS-OF-INITIALIZE.COMMS.1 ASV (COMPACT.UNIT)]
 #[Unit: INVISIBLE.PICTURE.2 ASV (COMPACT.UNIT)]
 #[Unit: VIEWPORT-LOAD.IRIS-FLAVORS-OF-INITIALIZE.COMMS.1 ASV (COMPACT.UNIT)]
 #[Unit: SOUTH.TANK ASV] #[Unit: NORTH.TANK ASV] #[Unit: WEST.TANK ASV]
 #[Unit: INVISIBLE.PICTURE.10 ASV])
VARIABLE.VALUES)
 (OPEN-ALL-IMAGES! (AI3-OPEN-ALL-IMAGES) METHOD (#[Unit: METHOD KEEDATATYPES]))
 (OPEN.ALL.PICTURES! (OPEN-ALL-VIEWPORTS) METHOD (#[Unit: METHOD KEEDATATYPES]))
 (VIEWPORT.STATUS.ON.KBLOAD NIL OVERRIDE.VALUES
((ONE.OF CREATE-WINDOWS ASK-TO-CREATE-WINDOWS DONT-CREATE-WINDOWS CREATE-CLOSED CRE
 ASK-TO-OPEN CREATE-OPEN))
(ASK-TO-CREATE-WINDOWS CREATE-SAME))
 ))

(NORTH.TANK
 ("goodpasturer" "9-14-87 14:08:01" "goodpasturer" "12-3-87 10:29:01")
 NIL
 ((BITMAPS KEEPICTURES))
 NIL
 0
 ((BACK.TRACK
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'PREVIOUS.POSITION)
              'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'INVERT)
  (PUT.VALUE THISUNIT
```

```
        'NODE.LOC
        (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'PREVIOUS.POSITION)))))
 (BITMAP (NEWTANK.NORTH.TANK.BITMAP.3812))
 (CONNECTING.LINES)
 (DEPENDENT.PICTURES)
 (DRIVER (#[Unit: ASV-DRIVER ASV]))
 (EAST.PATH (OBSTRUCTED))
 (HIGHLIGHTP (NIL))
 (LOCAL.COMPACT.UNIT.SLOTNAMES (NIL))
 (LOCATION-X ((AREF (GET.VALUE 'TANK1 'POSITION) 1)))
 (LOCATION-Y ((AREF (GET.VALUE 'TANK1 'POSITION) 2)))
 (MASK.BITMAP (NIL))
 (MOVE.EAST
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST) 'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST)
         'PREVIOUS.POSITION
         (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST)))))
 (MOVE.NORTH
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH) 'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH)
```

```
        'PREVIOUS.POSITION
        (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH)))))
 (MOVE.SOUTH
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH) 'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH)
        'PREVIOUS.POSITION
        (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH)))))
 (MOVE.WEST
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST) 'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST)
        'PREVIOUS.POSITION
        (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST)))))
 (NODE.LOC (CIRCLE.91))
 (NORTH.PATH (OBSTRUCTED))
 (OPAQUEP (T))
 (OPENP (OPEN))
 (POSITION (#S(POSITION :X 0.0 :Y -3.0)))
 (PREVIOUS.POSITION (#[Unit: CIRCLE.90 *Kb-?*]))
```

```
 (REGION (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 22 :HEIGHT 22)))
 (REGION.OF.SUBPICTURES (#S(REGION :LEFT -1.0 :BOTTOM 0.0 :WIDTH 23.0 :HEIGHT 23.0)))
 (SOUTH.PATH (OBSTRUCTED))
 (SUBPICTURES (#[Unit: SOUTH.TANK ASV]))
 (SUPERPICTURE (#[Unit: WEST.TANK ASV]))
 (TERRAIN (#[Unit: TERRAIN2 ASV]))
 (VIEWPORTS)
 (WEST.PATH (OBSTRUCTED))
 ))

(SOUTH.TANK
 ("goodpasturer" "9-14-87 14:20:59" "goodpasturer" "12-3-87 10:29:01")
 NIL
 ((BITMAPS KEEPICTURES))
 NIL
 ()
 ((BACK.TRACK
 ((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'PREVIOUS.POSITION)
              'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'INVERT)
  (PUT.VALUE THISUNIT
         'NODE.LOC
         (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'PREVIOUS.POSITION)))))
 (BITMAP (NEWTANK.SOUTH.TANK.BITMAP.3810))
 (CONNECTING.LINES)
 (DEPENDENT.PICTURES)
 (DRIVER (#[Unit: ASV-DRIVER ASV]))
 (EAST.PATH (OBSTRUCTED))
 (HIGHLIGHTP (NIL))
```

```
 (LOCAL.COMPACT.UNIT.SLOTNAMES (NIL))
 (LOCATION-X ((AREF (GET.VALUE 'TANK1 'POSITION) 1)))
 (LOCATION-Y ((AREF (GET.VALUE 'TANK1 'POSITION) 2)))
 (MASK.BITMAP (NIL))
 (MOVE.EAST
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST) 'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST)
         'PREVIOUS.POSITION
         (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST)))))
 (MOVE.NORTH
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH) 'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH)
         'PREVIOUS.POSITION
         (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH)))))
 (MOVE.SOUTH
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH) 'POSITION)
```

```
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH)
         'PREVIOUS.POSITION
         (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH)))))
 (MOVE.WEST
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST) 'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST)
         'PREVIOUS.POSITION
         (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST)))))
 (NODE.LOC (CIRCLE.91))
 (NORTH.PATH (OBSTRUCTED))
 (OPAQUEP (T))
 (OPENP (OPEN))
 (POSITION (#S(POSITION :X -1.0 :Y 1.0)))
 (PREVIOUS.POSITION (#[Unit: CIRCLE.90 *Kb-?*]))
 (REGION (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 22 :HEIGHT 22)))
 (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 22 :HEIGHT 22)))
 (SOUTH.PATH (OBSTRUCTED))
 (SUBPICTURES)
 (SUPERPICTURE (#[Unit: NORTH.TANK ASV]))
 (TERRAIN (#[Unit: TERRAIN2 ASV]))
 (VIEWPORTS)
 (WEST.PATH (OBSTRUCTED))
```

```
 ))

(WEST.TANK
 ("goodpasturer" "9-14-87 13:44:03" "goodpasturer" "12-3-87 10:29:01")
 NIL
 ((BITMAPS KEEPICTURES))
 NIL
 ()
 ((BACK.TRACK
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'PREVIOUS.POSITION)
              'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'INVERT)
  (PUT.VALUE THISUNIT
         'NODE.LOC
         (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'PREVIOUS.POSITION)))))
 (BITMAP (NEWTANK.WEST.TANK.BITMAP.3795))
 (CONNECTING.LINES)
 (DEPENDENT.PICTURES)
 (DRIVER (#[Unit: ASV-DRIVER ASV]))
 (EAST.PATH (OBSTRUCTED))
 (HIGHLIGHTP (NIL))
 (LOCAL.COMPACT.UNIT.SLOTNAMES (NIL))
 (LOCATION-X ((AREF (GET.VALUE 'TANK1 'POSITION) 1)))
 (LOCATION-Y ((AREF (GET.VALUE 'TANK1 'POSITION) 2)))
 (MASK.BITMAP (NIL))
 (MOVE.EAST
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
```

```
      (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST) 'POSITION)
      T
      NIL
      50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST)
       'PREVIOUS.POSITION
       (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST)))))
 (MOVE.NORTH
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
      'MOVE!
      (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH) 'POSITION)
      T
      NIL
      50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH)
       'PREVIOUS.POSITION
       (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH)))))
 (MOVE.SOUTH
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
      'MOVE!
      (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH) 'POSITION)
      T
      NIL
      50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH)
       'PREVIOUS.POSITION
       (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH)))))
```

```
 (MOVE.WEST
((LAMBDA (THISUNIT)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST) 'POSITION)
        T
        NIL
        50)
  (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
  (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST)
         'PREVIOUS.POSITION
         (GET.VALUE THISUNIT 'NODE.LOC))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST)))))
 (NODE.LOC (CIRCLE.91))
 (NORTH.PATH (OBSTRUCTED))
 (OPAQUEP (T))
 (OPENP (OPEN))
 (POSITION (#S(POSITION :X 2.0 :Y -2.0)))
 (PREVIOUS.POSITION (#[Unit: CIRCLE.90 *Kb-?*]))
 (REGION (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 22 :HEIGHT 22)))
 (REGION.OF.SUBPICTURES (#S(REGION :LEFT -1.0 :BOTTOM -3.0 :WIDTH 23.0 :HEIGHT 25.0)))
 (SOUTH.PATH (OBSTRUCTED))
 (SUBPICTURES (#[Unit: NORTH.TANK ASV]))
 (SUPERPICTURE (#[Unit: ASV ASV]))
 (TERRAIN (#[Unit: TERRAIN2 ASV]))
 (VIEWPORTS)
 (WEST.PATH (OBSTRUCTED))
 ))

(GOAL.LABEL
 ("goodpasturer" "9-11-87 10:18:16" "goodpasturer" "12-3-87 10:29:00")
 NIL
 ((BITMAPS KEEPICTURES))
 NIL
 0
```

```
 ((BITMAP (NEWTANK.GOAL.LABEL.BITMAP.3886))
  (EAST.PATH (OBSTRUCTED))
  (HIGHLIGHTP (NIL))
  (LABEL ("GOAL"))
  (LABEL.BITMAP (ASV.GOAL.LABEL.LABEL.BITMAP.3525))
  (LABEL.FONT)
  (LABEL.POSITION)
  (LOCAL.COMPACT.UNIT.SLOTNAMES (NIL))
  (MASK.BITMAP (NIL))
  (NORTH.PATH (OBSTRUCTED))
  (OPAQUEP (NIL))
  (OPENP (OPEN))
  (POSITION (#S(POSITION :X 100 :Y 300)))
  (PREVIOUS.POSITION (#[Unit: CIRCLE.12 *Kb-?*]))
  (REGION (#S(REGION :LEFT 0 :BOTTOM -16 :WIDTH 40 :HEIGHT 21)))
  (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM -16 :WIDTH 40 :HEIGHT 21)))
  (SCALE.FACTOR.X)
  (SCALE.FACTOR.Y)
  (SOUTH.PATH (OBSTRUCTED))
  (SUPERPICTURE (#[Unit: INVISIBLE.PICTURE.10 ASV]))
  (WEST.PATH (OBSTRUCTED))
  ))

(INVISIBLE.PICTURE.10
  ("goodpasturer" "9-10-87 13:52:39")
  NIL
  ((INVISIBLE.PICTURES KEEPICTURES))
  NIL
  ()
  ((OPENP (OPEN))
   (POSITION (#S(POSITION :X 0 :Y 0)))
   (REGION (NIL))
   (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 284 :WIDTH 140 :HEIGHT 188)))
   (SUBPICTURES (#[Unit: ASV ASV] #[Unit: GOAL.LABEL ASV]))
   (VIEWPORTS)
```

```
))

(ASV-DRIVER
 ("goodpasturer" "9-11-87 10:16:09" "goodpasturer" "11-12-87 13:38:41")
 NIL
 (DRIVERS)
 NIL
 ()
 ((RELATIVE.LOCATION (NOT.AT.GOAL))
  (RELATIVE.LOCATION.X (AT.GOAL.X))
  (RELATIVE.LOCATION.Y (AT.GOAL.Y))
  (RELATIVE.POSITION (AT.GOAL))
  (RULE.SET (#[Unit: ASV-DRIVER.RULES ASV]))
  (VEHICLE (#[Unit: ASV ASV]))
  (VISION
 ((LAMBDA (THISUNIT &AUX VEH TER GOALLOC VEHLOC)
   (SETQ VEH (GET.VALUE THISUNIT 'VEHICLE))
   (SETQ TER (GET.VALUE VEH 'TERRAIN))
   (SETQ GOALLOC
        (LIST (EVAL (GET.VALUE TER 'GOAL.LOCATION-X))
            (EVAL (GET.VALUE TER 'GOAL.LOCATION-Y))))
   (SETQ VEHLOC (LIST (EVAL (GET.VALUE VEH 'LOCATION-X)) (EVAL (GET.VALUE VEH 'LOCATION-Y))))
   (COND ((> (ABS (- (CAR VEHLOC) (CAR GOALLOC))) (ABS (- (CADR VEHLOC) (CADR GOALLOC))))
        (PUT.VALUE 'DIRECTIONS 'BEST.CHOICE 'EAST.WEST))
       (T (PUT.VALUE 'DIRECTIONS 'BEST.CHOICE 'NORTH.SOUTH)))
   (COND ((< (CAR VEHLOC) (CAR GOALLOC))
        (PUT.VALUE THISUNIT 'RELATIVE.LOCATION.X 'WEST.OF.GOAL))
       ((> (CAR VEHLOC) (CAR GOALLOC))
        (PUT.VALUE THISUNIT 'RELATIVE.LOCATION.X 'EAST.OF.GOAL))
       (T (PUT.VALUE THISUNIT 'RELATIVE.LOCATION.X 'AT.GOAL.X)))
   (COND ((> (CADR VEHLOC) (CADR GOALLOC))
        (PUT.VALUE THISUNIT 'RELATIVE.LOCATION.Y 'NORTH.OF.GOAL))
       ((< (CADR VEHLOC) (CADR GOALLOC))
        (PUT.VALUE THISUNIT 'RELATIVE.LOCATION.Y 'SOUTH.OF.GOAL))
       (T (PUT.VALUE THISUNIT 'RELATIVE.LOCATION.Y 'AT.GOAL.Y)))
```

```
(COND ((AND (EQUAL (GET.VALUE THISUNIT 'RELATIVE.LOCATION.Y) 'AT.GOAL.Y)
        (EQUAL (GET.VALUE THISUNIT 'RELATIVE.LOCATION.X) 'AT.GOAL.X))
    (PUT.VALUE THISUNIT 'RELATIVE.LOCATION 'AT.GOAL))
   (T (PUT.VALUE THISUNIT 'RELATIVE.LOCATION 'NOT.AT.GOAL)))
(LET ((WEST.STATUS (GET.VALUE (GET.VALUE VEH 'NODE.LOC) 'WEST))
   (EAST.STATUS (GET.VALUE (GET.VALUE VEH 'NODE.LOC) 'EAST))
   (NORTH.STATUS (GET.VALUE (GET.VALUE VEH 'NODE.LOC) 'NORTH))
   (SOUTH.STATUS (GET.VALUE (GET.VALUE VEH 'NODE.LOC) 'SOUTH)))
 (COND ((EQUAL WEST.STATUS NIL)
     (PUT.VALUE (GET.VALUE 'ASV 'NODE.LOC) 'WEST.PATH 'OBSTRUCTED))
     ((EQUAL (GET.VALUE WEST.STATUS 'HIGHLIGHTP) NIL)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'WEST.PATH 'CLEAR))
     ((EQUAL (GET.VALUE WEST.STATUS 'HIGHLIGHTP) 'GRAY)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'WEST.PATH 'EXPLORED))
     (T (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'WEST.PATH 'OBSTRUCTED)))
 (COND ((EQUAL EAST.STATUS NIL)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'EAST.PATH 'OBSTRUCTED))
     ((EQUAL (GET.VALUE EAST.STATUS 'HIGHLIGHTP) NIL)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'EAST.PATH 'CLEAR))
     ((EQUAL (GET.VALUE EAST.STATUS 'HIGHLIGHTP) 'GRAY)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'EAST.PATH 'EXPLORED))
     (T (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'EAST.PATH 'OBSTRUCTED)))
 (COND ((EQUAL NORTH.STATUS NIL)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'NORTH.PATH 'OBSTRUCTED))
     ((EQUAL (GET.VALUE NORTH.STATUS 'HIGHLIGHTP) NIL)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'NORTH.PATH 'CLEAR))
     ((EQUAL (GET.VALUE NORTH.STATUS 'HIGHLIGHTP) 'GRAY)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'NORTH.PATH 'EXPLORED))
     (T (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'NORTH.PATH 'OBSTRUCTED)))
 (COND ((EQUAL SOUTH.STATUS NIL)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'SOUTH.PATH 'OBSTRUCTED))
     ((EQUAL (GET.VALUE SOUTH.STATUS 'HIGHLIGHTP) NIL)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'SOUTH.PATH 'CLEAR))
     ((EQUAL (GET.VALUE SOUTH.STATUS 'HIGHLIGHTP) 'GRAY)
     (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'SOUTH.PATH 'EXPLORED))
```

```
        (T (PUT.VALUE (GET.VALUE VEH 'NODE.LOC) 'SOUTH.PATH 'OBSTRUCTED)))))))
  ))

(ASV
 ("goodpasturer" "9-11-87 10:16:09" "goodpasturer" "12-3-87 10:29:06")
 NIL
 ((BITMAPS KEEPICTURES) VEHICLES)
 NIL
 ()
 ((BACK.TRACK
((LAMBDA (THISUNIT)
  (SETQ BACKTRACK T)
  (LET ((DIRECTION (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'PREVIOUS.DIRECTION)))
   (CASE DIRECTION
    (WEST (UNITMSG THISUNIT 'MOVE.WEST))
    (EAST (UNITMSG THISUNIT 'MOVE.EAST))
    (SOUTH (UNITMSG THISUNIT 'MOVE.SOUTH))
    (NORTH (UNITMSG THISUNIT 'MOVE.NORTH))))
  (SETQ BACKTRACK NIL)))
 METHOD (#[Unit: METHOD KEEDATATYPES]) NIL ((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  (BITMAP (NEWTANK.TANK1.BITMAP.3799))
  (DRIVER (#[Unit: ASV-DRIVER ASV]))
  (EAST.PATH (OBSTRUCTED))
  (HIGHLIGHTP (NIL))
  (LABEL)
  (LABEL.BITMAP)
  (LABEL.FONT)
  (LABEL.POSITION)
  (LOCAL.COMPACT.UNIT.SLOTNAMES (NIL))
  (LOCATION-X ((AREF (GET.VALUE 'ASV 'POSITION) 0)))
  (LOCATION-Y ((AREF (GET.VALUE 'ASV 'POSITION) 1)))
  (MASK.BITMAP (NIL))
  (MOVE.EAST
((LAMBDA (THISUNIT)
  (UNITMSG 'TALK 'STOP)
```

```
  (UNITMSG 'TALK 'ACCELERATE-X)
  (UNITMSG THISUNIT 'CLOSE! NIL T)
  (UNITMSG THISUNIT 'OPEN! 1 T)
  (UNITMSG THISUNIT 'REDISPLAY!)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST) 'POSITION)
        T
        NIL
        50)
  (COND (BACKTRACK (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'INVERT))
     (T (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
        (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST)
              'PREVIOUS.POSITION
              (GET.VALUE THISUNIT 'NODE.LOC))
        (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST)
              'PREVIOUS.DIRECTION
              'WEST)))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'EAST))))
METHOD (#[Unit: METHOD KEEDATATYPES]))
 (MOVE.NORTH
((LAMBDA (THISUNIT)
  (UNITMSG 'TALK 'STOP)
  (UNITMSG 'TALK 'ACCELERATE-Y)
  (UNITMSG THISUNIT 'CLOSE! NIL T)
  (UNITMSG THISUNIT 'OPEN! 2 T)
  (UNITMSG THISUNIT 'REDISPLAY!)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH) 'POSITION)
        T
        NIL
        50)
  (COND (BACKTRACK (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'INVERT))
     (T (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
```

```
        (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH)
                'PREVIOUS.POSITION
                (GET.VALUE THISUNIT 'NODE.LOC))
        (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH)
                'PREVIOUS.DIRECTION
                'SOUTH)))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'NORTH))))
METHOD (#[Unit: METHOD KEEDATATYPES]))
 (MOVE.SOUTH
((LAMBDA (THISUNIT)
  (UNITMSG 'TALK 'STOP)
  (UNITMSG 'TALK 'DECELERATE-Y)
  (UNITMSG THISUNIT 'CLOSE! NIL T)
  (UNITMSG THISUNIT 'OPEN! 3 T)
  (UNITMSG THISUNIT 'REDISPLAY!)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH) 'POSITION)
        T
        NIL
        50)
  (COND (BACKTRACK (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'INVERT))
      (T (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
        (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH)
                'PREVIOUS.POSITION
                (GET.VALUE THISUNIT 'NODE.LOC))
        (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH)
                'PREVIOUS.DIRECTION
                'NORTH)))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'SOUTH))))
METHOD (#[Unit: METHOD KEEDATATYPES]))
 (MOVE.WEST
((LAMBDA (THISUNIT)
  (UNITMSG 'TALK 'STOP)
  (UNITMSG 'TALK 'DECELERATE-X)
```

```
  (UNITMSG THISUNIT 'CLOSE.SUBPICTURES!)
  (UNITMSG THISUNIT
        'MOVE!
        (GET.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST) 'POSITION)
        T
        NIL
        50)
  (COND (BACKTRACK (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'INVERT))
      (T (PUT.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'HIGHLIGHTP 'GRAY)
        (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST)
              'PREVIOUS.POSITION
              (GET.VALUE THISUNIT 'NODE.LOC))
        (PUT.VALUE (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST)
              'PREVIOUS.DIRECTION
              'EAST)))
  (PUT.VALUE THISUNIT 'NODE.LOC (GET.VALUE (GET.VALUE THISUNIT 'NODE.LOC) 'WEST))))
 METHOD (#[Unit: METHOD KEEDATATYPES]))
  (NODE.LOC (CIRCLE.91) NIL NIL NIL ((CARDINALITY.MIN (0)) (CARDINALITY.MAX (100))))
  (NORTH.PATH (OBSTRUCTED))
  (OPAQUEP (NIL))
  (OPENP (OPEN))
  (POSITION (#S(POSITION :X 0 :Y 450)))
  (PREVIOUS.POSITION (#[Unit: CIRCLE.90 *Kb-?*]))
  (REGION (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 22 :HEIGHT 22)))
  (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0.0 :BOTTOM -5.0 :WIDTH 24.0 :HEIGHT 27.0)))
  (SCALE.FACTOR.X)
  (SCALE.FACTOR.Y)
  (SOUTH.PATH (OBSTRUCTED))
  (SUBPICTURES (#[Unit: WEST.TANK ASV]))
  (SUPERPICTURE (#[Unit: INVISIBLE.PICTURE.10 ASV]))
  (TERRAIN (#[Unit: TERRAIN2 ASV]))
  (WEST.PATH (OBSTRUCTED))
  ))

(ASV-DRIVER.RULES
```

```
 ("goodpasturer" "9-10-87 14:09:56" "GOODPASTURER" "11-12-87 10:07:33")
 (DRIVING.RULES)
 ((CLASSES GENERICUNITS))
 NIL
 ((ASSERTION)
  (EXTERNAL.FORM)
  (PREMISE)
  )
 ((EXTERNAL.FORM (NIL))
  ))

(TERRAIN2
 ("goodpasturer" "9-11-87 10:16:09" "goodpasturer" "11-12-87 13:38:41")
 (TERRAIN.EXAMPLES)
 ((CLASSES GENERICUNITS))
 NIL
 0
 ((GOAL (CIRCLE.63))
  (GOAL.LOCATION-X ((AREF (GET.VALUE (GET.VALUE 'TERRAIN2 'GOAL) 'POSITION) 0)))
  (GOAL.LOCATION-Y ((AREF (GET.VALUE (GET.VALUE 'TERRAIN2 'GOAL) 'POSITION) 1)))
  ))

(FIND.WEST
 ("goodpasturer" "9-10-87 13:50:11" "goodpasturer" "12-3-87 10:34:15")
 NIL
 (INIT.DIRECTIONS)
 NIL
 0
 ((EXTERNAL.FORM ((IF (?CIRCLE IS IN CIRCLES) THEN (LISP (UNITMSG ?CIRCLE 'FIND.WEST)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(FIND.NORTH
```

```
 ("goodpasturer" "9-10-87 13:50:11" "goodpasturer" "12-3-87 10:34:15")
 NIL
 (INIT.DIRECTIONS)
 NIL
 0
 ((EXTERNAL.FORM ((IF (?CIRCLE IS IN CIRCLES) THEN (LISP (UNITMSG ?CIRCLE 'FIND.NORTH)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(FIND.SOUTH
 ("goodpasturer" "9-10-87 13:50:11" "goodpasturer" "12-3-87 10:34:14")
 NIL
 (INIT.DIRECTIONS)
 NIL
 0
 ((EXTERNAL.FORM ((IF (?CIRCLE IS IN CIRCLES) THEN (LISP (UNITMSG ?CIRCLE 'FIND.SOUTH)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(FIND.EAST
 ("goodpasturer" "9-10-87 13:50:11" "goodpasturer" "12-3-87 10:34:14")
 NIL
 (INIT.DIRECTIONS)
 NIL
 0
 ((EXTERNAL.FORM ((IF (?CIRCLE IS IN CIRCLES) THEN (LISP (UNITMSG ?CIRCLE 'FIND.EAST)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))
```

```
(INIT.DIRECTIONS
 ("goodpasturer" "9-10-87 13:50:11" "goodpasturer" "9-22-87 9:35:23")
 (INIT.WORLD.RULES)
 ((CLASSES GENERICUNITS))
 NIL
 ((ASSERTION)
  (EXTERNAL.FORM)
  (PREMISE)
  )
 ())

(INIT.NODES
 ("goodpasturer" "9-10-87 13:50:11" "goodpasturer" "12-3-87 10:34:15")
 NIL
 (INIT.WORLD.RULES)
 NIL
 ()
 ((EXTERNAL.FORM ((IF (?CIRCLE IS IN CIRCLES) THEN (LISP (ADD.PARENT ?CIRCLE 'NODES 'MEMBER T))
)
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(INIT.WORLD.RULES
 ("goodpasturer" "9-10-87 13:50:11" "goodpasturer" "9-10-87 12:50:13")
 ((RULES RULESYSTEM3))
 ((RULE.CLASSES RULESYSTEM3))
 NIL
 ((ASSERTION)
  (EXTERNAL.FORM)
  (PREMISE)
  (STEP.BY (BY.PREMISE))
  )
 ())
```

```
(TERRAIN1
 ("goodpasturer" "9-11-87 10:16:09" "goodpasturer" "9-11-87 9:16:10")
 NIL
 (TERRAIN.EXAMPLES)
 NIL
 ()
 ((GOAL.LOCATION (50) NIL (#[Unit: INTEGER KEEDATATYPES]) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  (GOAL.LOCATION-X (0))
  (GOAL.LOCATION-Y (50))
  (SIZE NIL NIL NIL NIL ((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  (TYPE (FLAT))
  ))

(DRIVING.RULES
 ("goodpasturer" "9-10-87 14:09:56" "goodpasturer" "9-10-87 13:09:58")
 ((RULES RULESYSTEM3))
 ((RULE.CLASSES RULESYSTEM3))
 NIL
 ((ASSERTION)
  (EXTERNAL.FORM)
  (PREMISE)
  (STEP.BY (BY.PREMISE))
  )
 ())

(TERRAIN.EXAMPLES
 ("goodpasturer" "9-11-87 10:16:09" "goodpasturer" "9-11-87 9:16:09")
 (ELEMENTS)
 ((CLASSES GENERICUNITS))
 NIL
 ((GOAL.LOCATION-X NIL OVERRIDE.VALUES NIL NIL ((CARDINALITY.MIN (0)) (CARDINALITY.MAX
  (GOAL.LOCATION-Y NIL OVERRIDE.VALUES NIL NIL ((CARDINALITY.MIN (0)) (CARDINALITY.MAX
  (SIZE NIL NIL (#[Unit: INTEGER KEEDATATYPES]) NIL
((CARDINALITY.MIN (0)) (CARDINALITY.MAX (100))))
```

```
 (TYPE NIL NIL ((ONE.OF FLAT ROLLING RUGGED)) NIL ((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
 )
 0)

(DRIVERS
 ("goodpasturer" "9-11-87 10:16:09" "goodpasturer" "9-22-87 8:59:40")
 (ELEMENTS)
 ((CLASSES GENERICUNITS))
 NIL
 ((RELATIVE.LOCATION NIL NIL ((ONE.OF NOT.AT.GOAL AT.GOAL)) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  (RELATIVE.LOCATION.X NIL OVERRIDE.VALUES ((ONE.OF EAST.OF.GOAL WEST.OF.GOAL AT.GOAL.X)) N
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  (RELATIVE.LOCATION.Y NIL OVERRIDE.VALUES ((ONE.OF NORTH.OF.GOAL SOUTH.OF.GOAL AT.GOAL.Y
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  (RULE.SET NIL NIL (#[Unit: RULE.CLASSES RULESYSTEM3]))
  (SEEK.GOAL ((LAMBDA (THISUNIT) (ASSERT NIL 'DRIVING.RULES))) METHOD
(#[Unit: METHOD KEEDATATYPES]))
  (VEHICLE NIL NIL (#[Unit: VEHICLES ASV]) NIL ((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
  (VISION
((LAMBDA (THISUNIT &AUX VEH TER GOALLOC VEHLOC)
   (SETQ VEH (GET.VALUE THISUNIT 'VEHICLE))
   (SETQ TER (GET.VALUE VEH 'TERRAIN))
   (SETQ GOALLOC (LIST (GET.VALUE TER 'GOAL.LOCATION-X) (GET.VALUE TER 'GOAL.LOCATION-Y)))
   (SETQ VEHLOC (LIST (GET.VALUE VEH 'LOCATION-X) (GET.VALUE VEH 'LOCATION-Y)))
   (COND ((< (CAR VEHLOC) (CAR GOALLOC))
         (PUT.VALUE THISUNIT 'RELATIVE.LOCATION.X 'WEST.OF.GOAL))
        ((> (CAR VEHLOC) (CAR GOALLOC))
         (PUT.VALUE THISUNIT 'RELATIVE.LOCATION.X 'EAST.OF.GOAL))
        ((> (CADR VEHLOC) (CADR GOALLOC))
         (PUT.VALUE THISUNIT 'RELATIVE.LOCATION.Y 'NORTH.OF.GOAL))
        ((< (CADR VEHLOC) (CADR GOALLOC))
         (PUT.VALUE THISUNIT 'RELATIVE.LOCATION.Y 'SOUTH.OF.GOAL))
        (T (PUT.VALUE THISUNIT 'RELATIVE.LOCATION 'AT.GOAL)))))
METHOD (#[Unit: METHOD KEEDATATYPES]))
```

```
 )
 ())

(ELEMENTS
 ("goodpasturer" "9-11-87 10:16:09" "goodpasturer" "9-11-87 9:16:09")
 NIL
 ((CLASSES GENERICUNITS))
 NIL
 ()
 ())

(VEHICLES
 ("goodpasturer" "9-11-87 10:16:09" "goodpasturer" "9-11-87 9:16:09")
 (ELEMENTS)
 ((CLASSES GENERICUNITS))
 NIL
 ((DRIVER NIL NIL (#[Unit: DRIVERS ASV]) NIL ((CARDINALITY.MAX (1)) (CARDINALITY.MIN (1))))
  (LOCATION-X NIL OVERRIDE.VALUES NIL NIL ((CARDINALITY.MIN (0)) (CARDINALITY.MAX (100))
  (LOCATION-Y NIL OVERRIDE.VALUES NIL NIL ((CARDINALITY.MIN (0)) (CARDINALITY.MAX (100))
  (TERRAIN NIL NIL (#[Unit: TERRAIN.EXAMPLES ASV]) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
 )
 ())

(COMMANDERS
 ("goodpasturer" "9-11-87 10:20:50" "goodpasturer" "9-11-87 9:20:50")
 NIL
 ((CLASSES GENERICUNITS))
 NIL
 ()
 ())

(IRIS.COMMS
 ("goodpasturer" "12-2-87 13:00:35" "goodpasturer" "12-2-87 13:01:31")
 NIL
```

```
 ((CLASSES GENERICUNITS))
 NIL
 0
 0)

(BUILD.TERRAIN.MODEL
 ("goodpasturer" "9-11-87 10:19:36" "goodpasturer" "12-2-87 13:02:59")
 ((ENTITIES GENERICUNITS))
 ((CLASSES GENERICUNITS))
 NIL
 ((INIT.NODES
(LAMBDA (THISUNIT &AUX X Y)
 (DO ((Y 0 (+ Y 50)))
    ((= Y 500) NIL)
  (DO ((X 0 (+ X 50)))
     ((= X 500) NIL)
    (UNITMSG 'CIRCLES
          'CREATE.INSTANCE!
          'NEWTANK
          NIL
          'INVISIBLE.PICTURE.10
          (LIST X Y)
          NIL
          10
          1))))
 METHOD (#[Unit: METHOD KEEDATATYPES]))
  (INIT.VIEWPORT
(LAMBDA NIL (UNITMSG 'VIEWPORTS 'CREATE.INSTANCE! NIL 'TANK-WORLD '(50 50 500 500))) METHOD
(#[Unit: METHOD KEEDATATYPES]))
 )
 0)

(ASV.TERRAIN.RULES
 ("goodpasturer" "9-11-87 10:20:00" "goodpasturer" "12-3-87 10:34:04")
 NIL
```

```
 (SET.UP.TERRAIN.RULES)
 NIL
 0
 ((EXTERNAL.FORM
((IF (?CIRCLE IS IN CLASS NODES)
    (THE HIGHLIGHTP OF ?CIRCLE IS GRAY)
    THEN
    (LISP (PUT.VALUE ?CIRCLE 'HIGHLIGHTP 'NIL)))))
  (MAKE.AND.WORLD? (NIL))
  (PARSE.ERRORS)
  (RULE.TYPE (SAME.WORLD.ACTION))
  ))

(SET.UP.TERRAIN.RULES
 ("goodpasturer" "9-11-87 10:20:00" "goodpasturer" "12-2-87 13:04:16")
 ((RULES RULESYSTEM3))
 ((RULE.CLASSES RULESYSTEM3))
 NIL
 ((ASSERTION)
  (EXTERNAL.FORM)
  (PREMISE)
  )
 0)

(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 VIEWPORT-LOAD.IRIS-FLAVORS-OF-INITIALIZE.COMMS.1
 0
 ((AI3-SLOT-VIEWPORTS ACTIVEIMAGES3))
  0
  (
   (TITLE.FONT (*MEDIUM-BOLD-FONT*))
   (BORDERS (8))
   (IMAGE-CLASS (#[Unit: METHOD-ACTUATORS ACTIVEIMAGES3]))
   (SCROLL.ON.P (NIL))
   (MOUSELEFTFN! (METHOD-ACTUATOR))
```

```
    (OPENP (OPEN))
    (LISP.WINDOW.REGION (#S(REGION :LEFT 178 :BOTTOM 688 :WIDTH 250 :HEIGHT 60)))
    (ATTACHED-OBJECTS (#[Slot: LOAD.IRIS-FLAVORS INITIALIZE.COMMS ASV OWN]))
    (LISP.WINDOW (ASV.VIEWPORT-LOAD.IRIS-FLAVORS-OF-INITIALIZE.COMMS.1.WINDOW.4114))
    (MAPPING ((0 0 1 1)))
    (TITLE ("Initialize.Comms's Load.Iris-Flavors"))
    (VIEWED.PICTURE (#[Unit: INVISIBLE.PICTURE.2 ASV (COMPACT.UNIT)]))
    (CACHED.PICTURE.REGIONS
 ((#[Unit: BOX-STRING-LOAD.IRIS-FLAVORS-OF-INITIALIZE.COMMS.1 ASV (COMPACT.UNIT)] :OTHER
   . #S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 234 :HEIGHT 32))))
    (ACTUATOR (#[Unit: BOX-STRING-LOAD.IRIS-FLAVORS-OF-INITIALIZE.COMMS.1 ASV (COMPACT.UNIT)]))
    (AVPUT (METHOD-ACTUATOR-RESPONSE))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 BOX-STRING-LOAD.IRIS-FLAVORS-OF-INITIALIZE.COMMS.1
  ()
  ((AI3-BOX-STRINGS ACTIVEIMAGES3))
   ()
   (
    (BORDERS (0))
    (STRINGS (("ASV>INITIALIZE.COMMS::LOAD.IRIS-FLAVORS!method")))
    (WRAPP (NIL))
    (WIDTH (234))
    (OPENP (OPEN))
    (SUPERPICTURE (#[Unit: INVISIBLE.PICTURE.2 ASV (COMPACT.UNIT)]))
    (JUSTIFICATION.H (CENTER))
    (JUSTIFICATION.V (CENTER))
    (FONT (*MEDIUM-FONT*))
    (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 234 :HEIGHT 32)))
    (REGION (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 234 :HEIGHT 32)))
    (HEIGHT (32))
    (OPAQUEP (T))
    (POSITION (#S(POSITION :X 0 :Y 0)))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
```

```
 INVISIBLE.PICTURE.2
  0
  ((INVISIBLE.PICTURES KEEPICTURES))
   0
   (
    (VIEWPORTS (#[Unit: VIEWPORT-LOAD.IRIS-FLAVORS-OF-INITIALIZE.COMMS.1 ASV (COMPACT.UN
    (POSITION (#S(POSITION :X 0 :Y 0)))
    (REGION (NIL))
    (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 210 :HEIGHT 19)))
    (SUBPICTURES (#[Unit: BOX-STRING-LOAD.IRIS-FLAVORS-OF-INITIALIZE.COMMS.1 ASV (COMPACT
    (MOUSERIGHTFN! (KPOBJECT-MOUSEANYFN))
    (MOUSELEFTFN! (KPOBJECT-MOUSEANYFN))
    (OPENP (OPEN))
    (MOUSEMIDDLEFN! (PICTURE-MOVE-MOUSEFN))
))
(INITIALIZE.COMMS
  ("GOODPASTURER" "10-29-87 9:03:07" "goodpasturer" "12-2-87 13:18:06")
  NIL
  (IRIS.COMMS (CLASSES GENERICUNITS))
  NIL
  0
  ((INITIALIZE.COMMS ((LAMBDA (THISUNITT) (START-CON))) METHOD (#[Unit: METHOD KEEDATAT
 ((ACTIVEIMAGES3 (#[Unit: VIEWPORT-INITIALIZE.COMMS-OF-INITIALIZE.COMMS.2 ASV (COMPACT.
   . UNIQUE.VALUES)))
   (LOAD.IRIS-FLAVORS ((LAMBDA (THISUNITT) (LOAD 'IRIS2-FLAVOR.LISP))) METHOD
 (#[Unit: METHOD KEEDATATYPES]) NIL
 ((ACTIVEIMAGES3 (#[Unit: VIEWPORT-LOAD.IRIS-FLAVORS-OF-INITIALIZE.COMMS.1 ASV (COMPAC
   . UNIQUE.VALUES)))
   ))

(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 VIEWPORT-INITIALIZE.COMMS-OF-INITIALIZE.COMMS.2
  0
  ((AI3-SLOT-VIEWPORTS ACTIVEIMAGES3))
   0
```

```
  (
   (TITLE.FONT (*MEDIUM-BOLD-FONT*))
   (BORDERS (8))
   (IMAGE-CLASS (#[Unit: METHOD-ACTUATORS ACTIVEIMAGES3]))
   (SCROLL.ON.P (NIL))
   (MOUSELEFTFN! (METHOD-ACTUATOR))
   (OPENP (OPEN))
   (LISP.WINDOW.REGION (#S(REGION :LEFT 178 :BOTTOM 615 :WIDTH 250 :HEIGHT 60)))
   (ATTACHED-OBJECTS (#[Slot: INITIALIZE.COMMS INITIALIZE.COMMS ASV OWN]))
   (LISP.WINDOW (ASV.VIEWPORT-INITIALIZE.COMMS-OF-INITIALIZE.COMMS.2.WINDOW.4115))
   (MAPPING ((0 0 1 1)))
   (TITLE ("Initialize.Comms's Initialize.Comms"))
   (VIEWED.PICTURE (#[Unit: INVISIBLE.PICTURE.7 ASV (COMPACT.UNIT)]))
   (CACHED.PICTURE.REGIONS
((#[Unit: BOX-STRING-INITIALIZE.COMMS-OF-INITIALIZE.COMMS.2 ASV (COMPACT.UNIT)] :OTHER
  . #S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 234 :HEIGHT 32))))
   (ACTUATOR (#[Unit: BOX-STRING-INITIALIZE.COMMS-OF-INITIALIZE.COMMS.2 ASV (COMPACT.UNIT)]))
   (AVPUT (METHOD-ACTUATOR-RESPONSE))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 BOX-STRING-INITIALIZE.COMMS-OF-INITIALIZE.COMMS.2
 ()
 ((AI3-BOX-STRINGS ACTIVEIMAGES3))
  ()
  (
   (BORDERS (0))
   (STRINGS (("ASV>INITIALIZE.COMMS::INITIALIZE.COMMS!method")))
   (WRAPP (NIL))
   (WIDTH (234))
   (OPENP (OPEN))
   (SUPERPICTURE (#[Unit: INVISIBLE.PICTURE.7 ASV (COMPACT.UNIT)]))
   (JUSTIFICATION.H (CENTER))
   (JUSTIFICATION.V (CENTER))
   (FONT (*MEDIUM-FONT*))
   (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 234 :HEIGHT 32)))
```

```
   (HIGHLIGHTP (NIL))
   (REGION (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 234 :HEIGHT 32)))
   (HEIGHT (32))
   (OPAQUEP (T))
   (POSITION (#S(POSITION :X 0 :Y 0)))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 INVISIBLE.PICTURE.7
 ()
 ((INVISIBLE.PICTURES KEEPICTURES))
  ()
  (
   (VIEWPORTS (#[Unit: VIEWPORT-INITIALIZE.COMMS-OF-INITIALIZE.COMMS.2 ASV (COMPACT.UNI
   (POSITION (#S(POSITION :X 0 :Y 0)))
   (REGION (NIL))
   (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 209 :HEIGHT 23)))
   (SUBPICTURES (#[Unit: BOX-STRING-INITIALIZE.COMMS-OF-INITIALIZE.COMMS.2 ASV (COMPACT.
   (MOUSERIGHTFN! (KPOBJECT-MOUSEANYFN))
   (MOUSELEFTFN! (KPOBJECT-MOUSEANYFN))
   (OPENP (OPEN))
   (MOUSEMIDDLEFN! (PICTURE-MOVE-MOUSEFN))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 VIEWPORT-START-WALKER-OF-ASV-COMMANDER.3
 ()
 ((AI3-SLOT-VIEWPORTS ACTIVEIMAGES3))
  ()
  (
   (TITLE.FONT (*MEDIUM-BOLD-FONT*))
   (BORDERS (8))
   (IMAGE-CLASS (#[Unit: METHOD-ACTUATORS ACTIVEIMAGES3]))
   (SCROLL.ON.P (NIL))
   (MOUSELEFTFN! (METHOD-ACTUATOR))
   (OPENP (OPEN))
   (LISP.WINDOW.REGION (#S(REGION :LEFT 461 :BOTTOM 686 :WIDTH 242 :HEIGHT 60)))
```

```
   (ATTACHED-OBJECTS (#[Slot: START-WALKER ASV-COMMANDER ASV OWN]))
   (LISP.WINDOW (ASV.VIEWPORT-START-WALKER-OF-ASV-COMMANDER.3.WINDOW.4117))
   (MAPPING ((0 0 1 1)))
   (TITLE ("Asv-Commander's Start-Walker"))
   (VIEWED.PICTURE (#[Unit: INVISIBLE.PICTURE.8 ASV (COMPACT.UNIT)]))
   (CACHED.PICTURE.REGIONS
 ((#[Unit: BOX-STRING-START-WALKER-OF-ASV-COMMANDER.3 ASV (COMPACT.UNIT)] :OTHER
  . #S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 226 :HEIGHT 32))))
   (ACTUATOR (#[Unit: BOX-STRING-START-WALKER-OF-ASV-COMMANDER.3 ASV (COMPACT.UNIT)]))
   (AVPUT (METHOD-ACTUATOR-RESPONSE))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 BOX-STRING-START-WALKER-OF-ASV-COMMANDER.3
 0
 ((AI3-BOX-STRINGS ACTIVEIMAGES3))
  0
  (
   (BORDERS (0))
   (STRINGS (("ASV>ASV-COMMANDER::START-WALKER!method")))
   (WRAPP (NIL))
   (WIDTH (226))
   (OPENP (OPEN))
   (SUPERPICTURE (#[Unit: INVISIBLE.PICTURE.8 ASV (COMPACT.UNIT)]))
   (JUSTIFICATION.H (CENTER))
   (JUSTIFICATION.V (CENTER))
   (FONT (*MEDIUM-FONT*))
   (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 226 :HEIGHT 32)))
   (REGION (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 226 :HEIGHT 32)))
   (HEIGHT (32))
   (OPAQUEP (T))
   (POSITION (#S(POSITION :X 0 :Y 0)))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 INVISIBLE.PICTURE.8
 0
```

```
 ((INVISIBLE.PICTURES KEEPICTURES))
  0
  (
   (VIEWPORTS (#[Unit: VIEWPORT-START-WALKER-OF-ASV-COMMANDER.3 ASV (COMPACT.UNIT)]))
   (POSITION (#S(POSITION :X 0 :Y 0)))
   (REGION (NIL))
   (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 209 :HEIGHT 22)))
   (SUBPICTURES (#[Unit: BOX-STRING-START-WALKER-OF-ASV-COMMANDER.3 ASV (COMPACT.UNI
   (MOUSERIGHTFN! (KPOBJECT-MOUSEANYFN))
   (MOUSELEFTFN! (KPOBJECT-MOUSEANYFN))
   (OPENP (OPEN))
   (MOUSEMIDDLEFN! (PICTURE-MOVE-MOUSEFN))
))
(ASV-COMMANDER
 ("goodpasturer" "9-11-87 10:20:50" "goodpasturer" "12-2-87 13:18:47")
 NIL
 (COMMANDERS)
 NIL
 ()
 ((INITIALIZE-TERRAIN
 ((LAMBDA (THISUNIT INITPOS GOAL)
   (ASSERT NIL 'SET.UP.TERRAIN.RULES)
   (PUT.VALUE 'ASV 'POSITION (GET.VALUE INITPOS 'POSITION))
   (PUT.VALUE 'ASV 'NODE.LOC INITPOS)
   (PUT.VALUE 'TERRAIN2 'GOAL GOAL)
   (PUT.VALUE 'GOAL.LABEL 'POSITION (GET.VALUE (GET.VALUE 'TERRAIN2 'GOAL) 'POSITION))
   (PUT.VALUE 'ASV-DRIVER 'RELATIVE.LOCATION 'NOT.AT.GOAL)
   (INITIALIZE.KEEWORLDS)
   (UNITMSG 'TANK-WORLD 'REDISPLAY.HARD!)))
 METHOD (#[Unit: METHOD KEEDATATYPES]) NIL
 ((ACTIVEIMAGES3 (#[Unit: VIEWPORT-INITIALIZE-TERRAIN-OF-ASV-COMMANDER.4 ASV (COMPACT.
  . UNIQUE.VALUES)))
  (START-WALKER
 ((LAMBDA (THISUNIT)
   (SETQ BACKTRACK NIL)
```

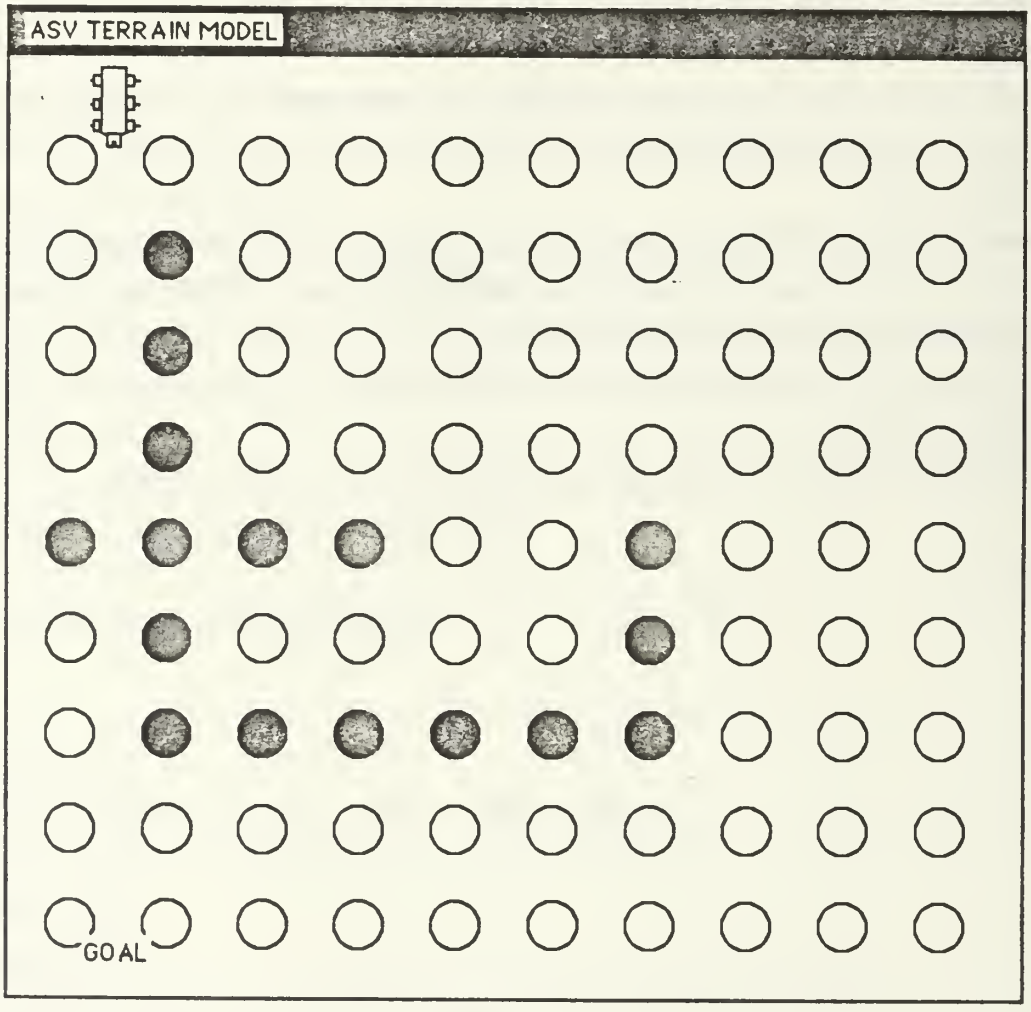
```
  (DO ((A 1 (+ 1 A)))
    ((EQUAL (GET.VALUE 'ASV-DRIVER 'RELATIVE.LOCATION) 'AT.GOAL) NIL)
   (UNITMSG 'ASV-DRIVER 'VISION)
   (UNITMSG 'ASV-DRIVER 'SEEK.GOAL))
  (UNITMSG 'TALK 'STOP)))
 METHOD (#[Unit: METHOD KEEDATATYPES]) NIL
((ACTIVEIMAGES3 (#[Unit: VIEWPORT-START-WALKER-OF-ASV-COMMANDER.3 ASV (COMPACT.UNIT)])
  . UNIQUE.VALUES)))
  ))

(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 VIEWPORT-INITIALIZE-TERRAIN-OF-ASV-COMMANDER.4
 ()
 ((AI3-SLOT-VIEWPORTS ACTIVEIMAGES3))
  ()
  (
   (TITLE.FONT (*MEDIUM-BOLD-FONT*))
   (BORDERS (8))
   (IMAGE-CLASS (#[Unit: METHOD-ACTUATORS ACTIVEIMAGES3]))
   (SCROLL.ON.P (NIL))
   (MOUSELEFTFN! (METHOD-ACTUATOR))
   (OPENP (OPEN))
   (LISP.WINDOW.REGION (#S(REGION :LEFT 460 :BOTTOM 615 :WIDTH 257 :HEIGHT 60)))
   (ATTACHED-OBJECTS (#[Slot: INITIALIZE-TERRAIN ASV-COMMANDER ASV OWN]))
   (LISP.WINDOW (ASV.VIEWPORT-INITIALIZE-TERRAIN-OF-ASV-COMMANDER.4.WINDOW.4118))
   (MAPPING ((0 0 1 1)))
   (TITLE ("Asv-Commander's Initialize-Terrain"))
   (VIEWED.PICTURE (#[Unit: INVISIBLE.PICTURE.9 ASV (COMPACT.UNIT)]))
   (CACHED.PICTURE.REGIONS
 ((#[Unit: BOX-STRING-INITIALIZE-TERRAIN-OF-ASV-COMMANDER.4 ASV (COMPACT.UNIT)] :OTHER
  . #S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 241 :HEIGHT 32))))
   (ACTUATOR (#[Unit: BOX-STRING-INITIALIZE-TERRAIN-OF-ASV-COMMANDER.4 ASV (COMPACT.UNIT)]))
   (AVPUT (METHOD-ACTUATOR-RESPONSE))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
```

```
 BOX-STRING-INITIALIZE-TERRAIN-OF-ASV-COMMANDER.4
  0
  ((AI3-BOX-STRINGS ACTIVEIMAGES3))
   0
   (
    (BORDERS (0))
    (STRINGS (("ASV>ASV-COMMANDER::INITIALIZE-TERRAIN!method")))
    (WRAPP (NIL))
    (WIDTH (241))
    (OPENP (OPEN))
    (SUPERPICTURE (#[Unit: INVISIBLE.PICTURE.9 ASV (COMPACT.UNIT)]))
    (JUSTIFICATION.H (CENTER))
    (JUSTIFICATION.V (CENTER))
    (FONT (*MEDIUM-FONT*))
    (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 241 :HEIGHT 32)))
    (REGION (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 241 :HEIGHT 32)))
    (HEIGHT (32))
    (OPAQUEP (T))
    (POSITION (#S(POSITION :X 0 :Y 0)))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 INVISIBLE.PICTURE.9
  0
  ((INVISIBLE.PICTURES KEEPICTURES))
   0
   (
    (VIEWPORTS (#[Unit: VIEWPORT-INITIALIZE-TERRAIN-OF-ASV-COMMANDER.4 ASV (COMPACT.UNI
    (POSITION (#S(POSITION :X 0 :Y 0)))
    (REGION (NIL))
    (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 227 :HEIGHT 32)))
    (SUBPICTURES (#[Unit: BOX-STRING-INITIALIZE-TERRAIN-OF-ASV-COMMANDER.4 ASV (COMPACT
    (MOUSERIGHTFN! (KPOBJECT-MOUSEANYFN))
    (MOUSELEFTFN! (KPOBJECT-MOUSEANYFN))
    (OPENP (OPEN))
    (MOUSEMIDDLEFN! (PICTURE-MOVE-MOUSEFN))
```

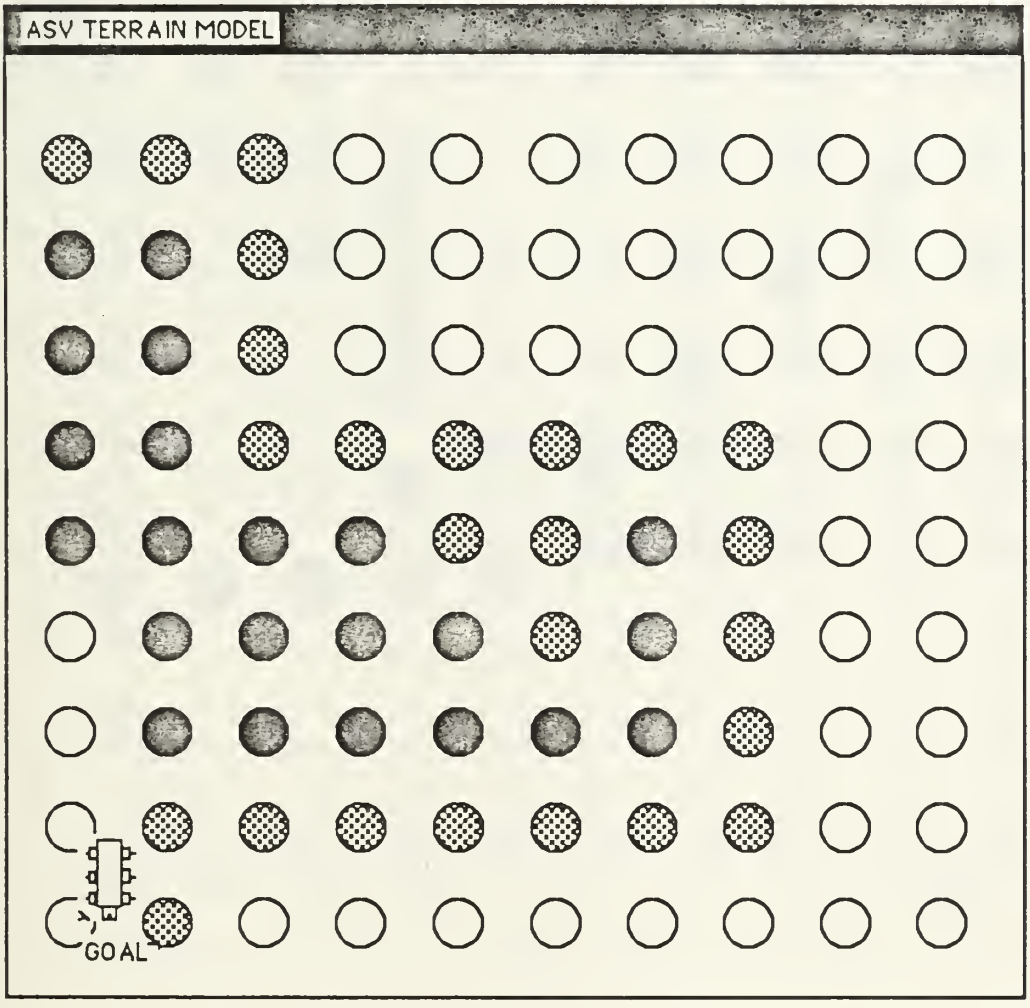
```
))
(TALK
 ("GOODPASTURER" "10-29-87 9:07:15" "goodpasturer" "12-2-87 13:30:10")
 NIL
 (IRIS.COMMS (CLASSES GENERICUNITS))
 NIL
 ()
 ((ACCELERATE-X ((LAMBDA (THISUNITT) (AX))) METHOD (#[Unit: METHOD KEEDATATYPES]) NIL
((ACTIVEIMAGES3 NIL . VARIABLE.VALUES)))
  (ACCELERATE-Y ((LAMBDA (THISUNITT) (AY))) METHOD (#[Unit: METHOD KEEDATATYPES]) NIL
((ACTIVEIMAGES3 NIL . VARIABLE.VALUES)))
  (DECELERATE-X ((LAMBDA (THISUNITT) (DX))) METHOD (#[Unit: METHOD KEEDATATYPES]) NIL
((ACTIVEIMAGES3 NIL . VARIABLE.VALUES)))
  (DECELERATE-Y ((LAMBDA (THISUNITT) (DY))) METHOD (#[Unit: METHOD KEEDATATYPES]) NIL
((ACTIVEIMAGES3 NIL . VARIABLE.VALUES)))
  (DISCONNECT.COMMS ((LAMBDA (THISUNITT) (H) (END-CON))) METHOD (#[Unit: METHOD KEEDATATYPE
NIL
((ACTIVEIMAGES3 (#[Unit: VIEWPORT-DISCONNECT.COMMS-OF-TALK.5 ASV (COMPACT.UNIT)])
  . UNIQUE.VALUES)
 (COMMENT "method")))
  (FORWARD.WAVE.GAIT ((LAMBDA (THISUNITT) (F))) METHOD (#[Unit: METHOD KEEDATATYPES]))
  (LISTEN ((LAMBDA (THISUNITT) (SEND TALK :GET-AN-OBJECT-FROM-IRIS)))
     METHOD
     (#[Unit: METHOD KEEDATATYPES])
     NIL
     ((ACTIVEIMAGES3 NIL . VARIABLE.VALUES)))
  (STOP ((LAMBDA (THISUNIIT) (S))) METHOD (#[Unit: METHOD KEEDATATYPES]))
  ))

(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 VIEWPORT-DISCONNECT.COMMS-OF-TALK.5
 0
 ((AI3-SLOT-VIEWPORTS ACTIVEIMAGES3))
  0
  (
```

```
   (TITLE.FONT (*MEDIUM-BOLD-FONT*))
   (BORDERS (8))
   (IMAGE-CLASS (#[Unit: METHOD-ACTUATORS ACTIVEIMAGES3]))
   (SCROLL.ON.P (NIL))
   (MOUSELEFTFN! (METHOD-ACTUATOR))
   (OPENP (OPEN))
   (LISP.WINDOW.REGION (#S(REGION :LEFT 347 :BOTTOM 546 :WIDTH 212 :HEIGHT 60)))
   (ATTACHED-OBJECTS (#[Slot: DISCONNECT.COMMS TALK ASV OWN]))
   (LISP.WINDOW (ASV.VIEWPORT-DISCONNECT.COMMS-OF-TALK.5.WINDOW.4129))
   (MAPPING ((0 0 1 1)))
   (TITLE ("Talk's Disconnect.Comms"))
   (VIEWED.PICTURE (#[Unit: INVISIBLE.PICTURE.11 ASV (COMPACT.UNIT)]))
   (CACHED.PICTURE.REGIONS
((#[Unit: BOX-STRING-DISCONNECT.COMMS-OF-TALK.5 ASV (COMPACT.UNIT)] :OTHER
  . #S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 196 :HEIGHT 32))))
   (ACTUATOR (#[Unit: BOX-STRING-DISCONNECT.COMMS-OF-TALK.5 ASV (COMPACT.UNIT)]))
   (AVPUT (METHOD-ACTUATOR-RESPONSE))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 BOX-STRING-DISCONNECT.COMMS-OF-TALK.5
 ()
 ((AI3-BOX-STRINGS ACTIVEIMAGES3))
  ()
  (
   (BORDERS (0))
   (STRINGS (("ASV>TALK::DISCONNECT.COMMS!method")))
   (WRAPP (NIL))
   (WIDTH (196))
   (OPENP (OPEN))
   (SUPERPICTURE (#[Unit: INVISIBLE.PICTURE.11 ASV (COMPACT.UNIT)]))
   (JUSTIFICATION.H (CENTER))
   (JUSTIFICATION.V (CENTER))
   (FONT (*MEDIUM-FONT*))
   (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 196 :HEIGHT 32)))
   (REGION (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 196 :HEIGHT 32)))
```

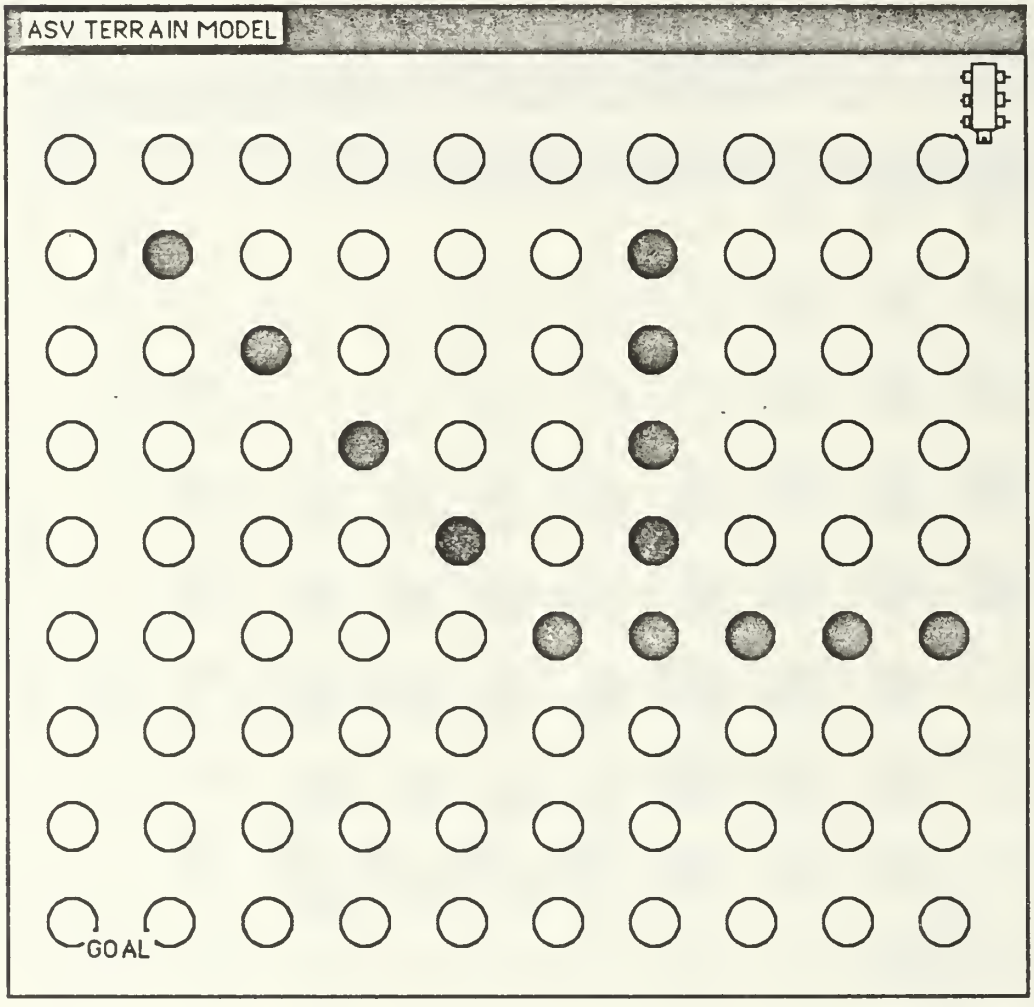
```
    (HEIGHT (32))
    (OPAQUEP (T))
    (POSITION (#S(POSITION :X 0 :Y 0)))
))
(ABSTRACT.UNITDEF.FORMAT.1 (COMPACT.UNIT)
 INVISIBLE.PICTURE.11
  ()
  ((INVISIBLE.PICTURES KEEPICTURES))
   ()
   (
    (VIEWPORTS (#[Unit: VIEWPORT-DISCONNECT.COMMS-OF-TALK.5 ASV (COMPACT.UNIT)]))
    (POSITION (#S(POSITION :X 0 :Y 0)))
    (REGION (NIL))
    (REGION.OF.SUBPICTURES (#S(REGION :LEFT 0 :BOTTOM 0 :WIDTH 235 :HEIGHT 24)))
    (SUBPICTURES (#[Unit: BOX-STRING-DISCONNECT.COMMS-OF-TALK.5 ASV (COMPACT.UNIT)]))
    (MOUSERIGHTFN! (KPOBJECT-MOUSEANYFN))
    (MOUSELEFTFN! (KPOBJECT-MOUSEANYFN))
    (OPENP (OPEN))
    (MOUSEMIDDLEFN! (PICTURE-MOVE-MOUSEFN))
))
KBEnd
```



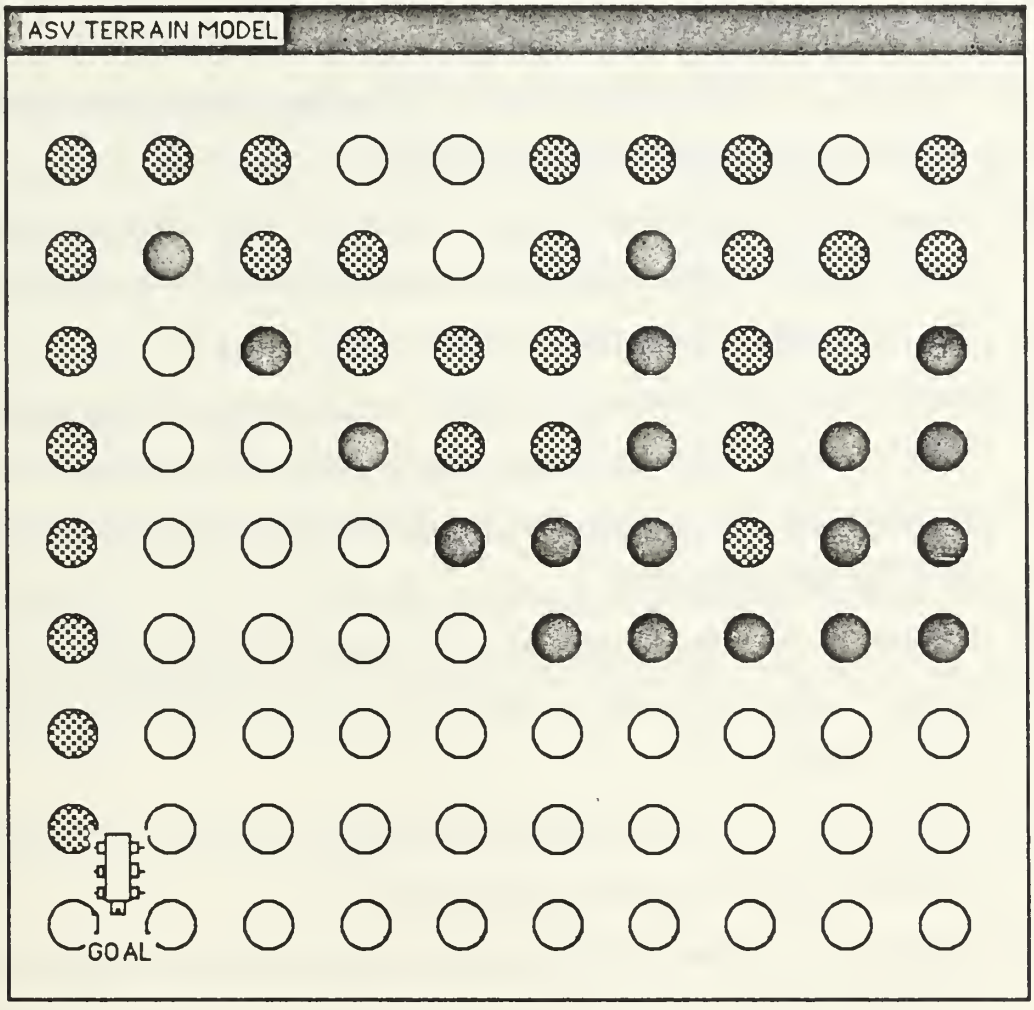
TERRAIN EXAMPLE 1



TERRAIN EXAMPLE 1 RESULTS



TERRAIN EXAMPLE 2



TERRAIN EXAMPLE 2 RESULTS

LIST OF REFERENCES

[1] *KEE Software Development System User's Manual (Version 3.0)*, Intellicorp, Mountain View, California, 1986 .

[2] Waldron, K. J. and McGhee, R. B., ''The Adaptive Suspension Vehicle,'' *IEEE Control Systems Magazine* , December 1986 .

[3] Lyman, R., *A Computer Simulation Study of Tripod Follow-the-Leader Gait Coordination for a Hexapod Walking Machine*, M. S. Thesis, Naval Postgraduate School, Monterey, California, June 1987.

[4] Giralt, G., Chatila, R., and Vaisset, M., ''An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots,'' in *Robotics Research: The First International Symposium*, ed. by Michael Brady and Richard Paul, The MIT Press, 1984 .

[5] McGhee, R. B. and Iswandhi, G. I., ''Adaptive Locomotion of a Multilegged Robot over Rough Terrain,'' *IEEE Transactions on Systems, Man, Cybernetics, SMC-9*, April 1979 .

[6] McGhee, R. B., ''Vehicular Legged Locomotion,'' in *Advances in Automation and Robotics*, ed. by G. N. Saridis, Jai Press, 1985 .

[7] Brooks, R. A., ''Planning Collision Free Motions for Pick and Place Operations,'' in *Robotics Research: The First International Symposium*, ed. by Michael Brady and Richard Paul, The MIT Press, 1984 .

[8] Richbourg, R. F., Rowe, N. C., Zyda, M. J., and McGhee, R. B., ''Solving Global, Two-dimensional Routing Problems Using Snell's Law and A* Search,'' *Proceedings of the IEEE International Conference on Robotics and Automation*, April 1987 .

[9] Mitchell, J. S. B. and Papadimtriou, C. H., *The Weighted Region Problem*, Technical Report, Department of Operations Research, Stanford University,

October 1985 (revised July 1986).

[10] Richbourg, R. F., *Solving a Class of Spatial Reasoning Problems: Minimal-Cost Path Planning in the Cartesian Plane*, Ph.D. dissertation, Naval Postgraduate School, Monterey, California, June 1987.

[11] Flynn, A. M., *Redundant Sensors for Mobile Robot Navigation*, Technical Report 859, Massachusetts Institute of Technology, October 1985.

[12] Kwak, S. H., *A Computer Simulation Study of a Free Gait Motion Coordination Algorithm for Rough-Terrain Locomotion by a Hexapod Walking Machine*, Ph.D. dissertation, The Ohio State University, Columbus, Ohio, August 1986.

[13] Lee, W. J., *A Computer Simulation Study of Omnidirectional Supervisory Control for Rough-Terrain Locomotion by a Multilegged Robot Vehicle*, Ph.D. dissertation, The Ohio State University, Columbus, Ohio, March 1984.

[14] *IRIS User's Guide* , Silicon Graphics, Inc. , Mountain View, California, 1986 .

[15] *ART Reference Manual*, Inference Corporation, Los Angeles, California, 1985 .

[16] *Texas Instrument Explorer User's Manual*, Texas Instrument Inc., Austin, Texas, 1985 .

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chief of Naval Operations Director, Information Systems (OP-945) Navy Department Washington, D.C. 20350-2000	1
4.	Department Chairman, Code 62 Department of Electronics and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
5.	Curriculum Officer, Code 32 Department of Electronics and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
6.	Professor Robert B. McGhee (Code 52Mz) Computer Science Department Naval Postgraduate School Monterey, California 93943-5000	15
7.	Professor Roberto Cristi (Code 62CX) Department of Electronics and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
8.	Department Chairman, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	1

9.	Professor D. L. Smith (Code 69Sm) Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	1
10.	Major Robert Richbourg USMA Office of Artificial Intelligence Analysis and Evaluation Attention: MADN-B Westpoint, New York 10996	1
11.	Prof. Kenneth J. Waldron Department of Mechanical Engineering Ohio State University 206 W. 18th Avenue Columbus, Ohio 43210	1
12.	Prof. C. A. Klein Department of Electrical Engineering Ohio State University 2015 Neil Ave. Columbus, Ohio 43210	1
13.	Dr. Wm. Isler DARPA/ISTO 1400 Wilson Blvd. Arlington, Virginia 22209	1
14.	Research Administration (Code 012) Naval Postgraduate School Monterey, California 93943	1